Shooting Trends using Geospatial Maps and Seasonal Modeling

Blake Tagget

10/1/2021

```
knitr::opts_chunk$set(echo = TRUE, fig.height=6, fig.width=10)
```

Data Exploration - New York City's Shooting Geography

Since this is neither a statistics course nor a sociology course, let's focus on seeing what neighborhood has the highest number of shooting victims and if there is an annual trend over the last few years. For the model we will quickly attempt to see if there is any seasonality in our incidents occurrence.

Disclaimers: - Since this project / report will live publicly on github and is for a course targeted at getting us to learn R and make reproducible reports as well as be thoughtful of communication, I am equally attempting to share the technical aspects of what makes this data analysis work as well as articulate interesting trends I see in the data. Finally, this only my second R file - so let's have some fun!

Gathering Some Data

There are two data sets we will download from the public internet: Historic NYC shooting data and NYC Geospatial polygon data

The first is the required historical shooting data set named "NYPD Shooting Incident Data (Historic)Public Safety." It has all the victims of shootings in NYC going back to 2006. You can get more information about it here: https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8

The second data set will allow use to create our geospatial graphs which draw boundaries for each neighborhood. While it's in a GeoJSON format for download, we quickly convert this to a Special Features object which can be manipulated in the same was as a usual data.frame. You can take a look at what the borough boundaries and neighborhoods in NYC are here: https://data.beta.nyc/dataset/0ff93d2d-90ba-457c-9f7e-39e47bf2ac5f/resource/35dd04fb-81b3-479b-a074-a27a37888ce7

Using these two data sets together, we will be able to graph interesting shooting metrics about each neighborhood. Fun fact, while we will use the Longitude and Latitude in the shooting data set, we will not leverage the *BORO* column as it has inconsistencies for 2020 (at least).

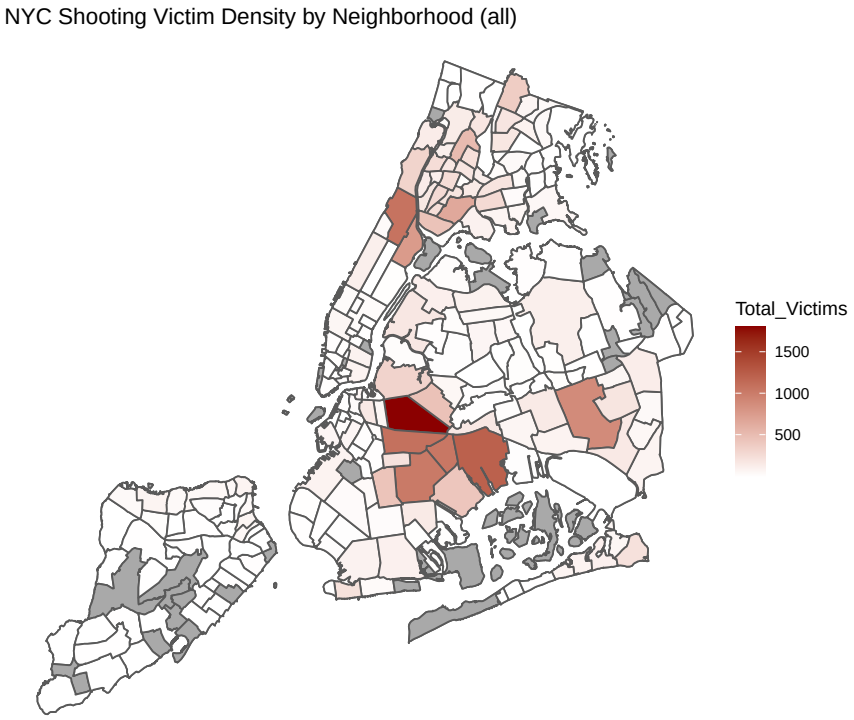
```
url_start <- ""
url_links <- c("https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD")
urls <- str_c(url_start,url_links)
shooting_data <- read_csv(urls[1]) %>%
  mutate(OCCUR_DATE=mdy(OCCUR_DATE),OCCUR_YR=year(OCCUR_DATE))

shooting_data_trimmed <- shooting_data %>%
  dplyr::select(OCCUR_YR,OCCUR_DATE,OCCUR_TIME,VIC_AGE_GROUP,VIC_SEX,VIC_RACE,Latitude,Longitude)
shooting_data_trimmed <- tibble::rowid_to_column(shooting_data_trimmed, "rowID")
```

```
r <- GET('https://data.beta.nyc/dataset/0ff93d2d-90ba-457c-9f7e-39e47bf2ac5f/resource/35dd04fb-81b3-479
nyc_neighborhoods <- readOGR(content(r,'text'), 'OGRGeoJSON', verbose = F)

nyc_neighborhoods_df <- tidy(nyc_neighborhoods)

nyc_neighborhoods_sf <- st_as_sf(nyc_neighborhoods)
nyc_neighborhoods_sf <- tibble::rowid_to_column(nyc_neighborhoods_sf, "neighborhood_code")

shooting_sf <- st_as_sf(shooting_data, coords = c("Longitude","Latitude"),crs= st_crs(nyc_neighborhoods
shooting_trimmed_sf <- st_as_sf(shooting_data_trimmed, coords = c("Longitude","Latitude"),crs= st_crs(ny

join_sf = st_join(nyc_neighborhoods_sf,shooting_trimmed_sf,join=st_intersects,left=TRUE)
```

High Level Analysis

Let's start off by looking at the entire shooting data set, count the total victims in each neighborhood, and produce a heat map to see if any particular neighborhood stands out. For visualization purposes, I have colored the neighborhoods without victims in gray and placed the shooting density on a scale from white (low victim count) to dark red (high victim count).

```
trimmed_join_1 <- join_sf %>%
  filter(!is.na(rowID)) %>%
  st_drop_geometry()

count_df <- dplyr::count(trimmed_join_1,neighborhood, name="Total_Victims")

join_sf_1 <- nyc_neighborhoods_sf %>%
  left_join(count_df)

plot_transform_1 <- ggplot() +
  geom_sf(data=nyc_neighborhoods_sf) +
  geom_sf(data=join_sf_1, aes(fill=Total_Victims)) +
  theme_void() +
  scale_fill_gradient2(high="darkred",na.value = 'darkgray') +
  ggtitle("NYC Shooting Victim Density by Neighborhood (all)")

plot_transform_1
```




```
sorted_sf <- arrange(join_sf_1,desc(Total_Victims))

total_shots <- sum(sorted_sf$Total_Victims,na.rm=T)
first_yr <- min(join_sf$OCCUR_YR,na.rm=T)
last_yr <- max(join_sf$OCCUR_YR,na.rm=T)
neighborhood <- sorted_sf$neighborhood[1]
shot_count <- sorted_sf$Total_Victims[1]
```

In summary, there are a lot of victims of shootings in NYC (23,673 between 2006 and 2020). The high level analysis here is that the neighborhood of Bedford-Stuyvesant has the most shooting victims with a total of 1,801 victims in that time frame.

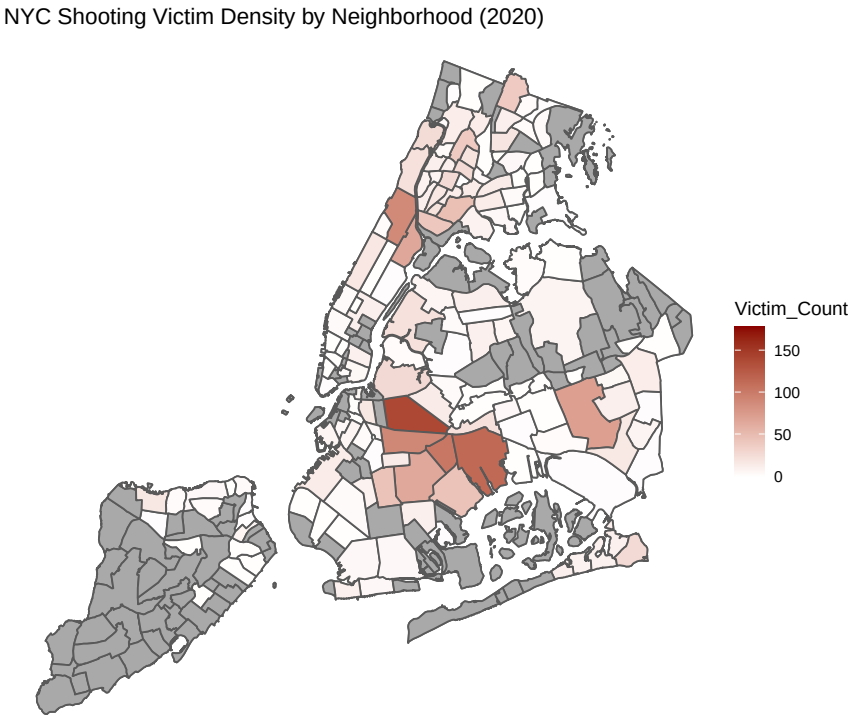
Annual Trend Analysis

Now let's look at the past few years and see if we can spot a trend of some sort for total victims by neighborhood by year.

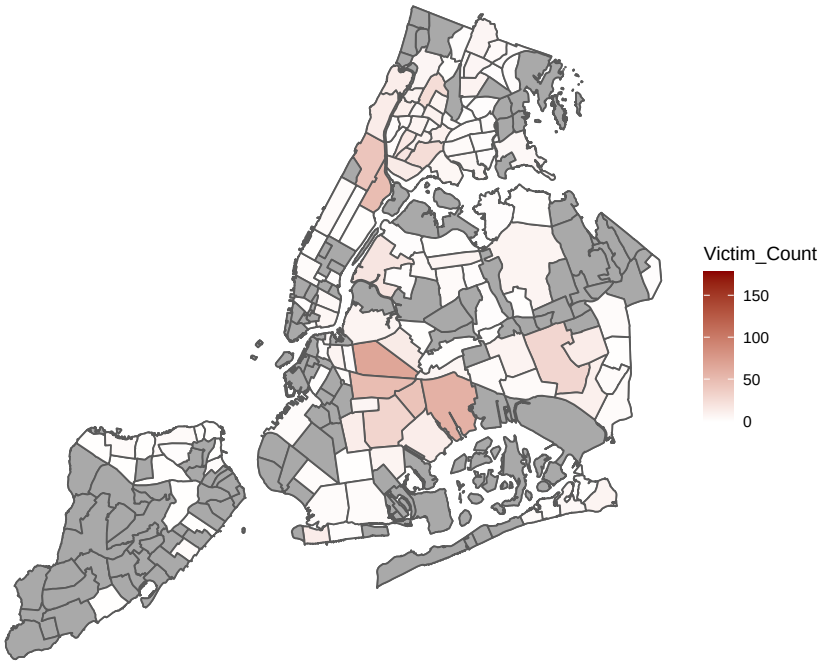
If you're Kniting this yourself, you should be able to enter in a number between 1 and 14 into the numberOfYears variable and produce all the graphs in order.

```
numberOfYears <- 4

trimmed_join_2 <- join_sf %>%
  filter(!is.na(rowID)) %>%
  st_drop_geometry()

count_df_2 <- dplyr::count(trimmed_join_2,neighborhood,OCCUR_YR)

max_range <- max(count_df_2$n,na.rm = T)
```

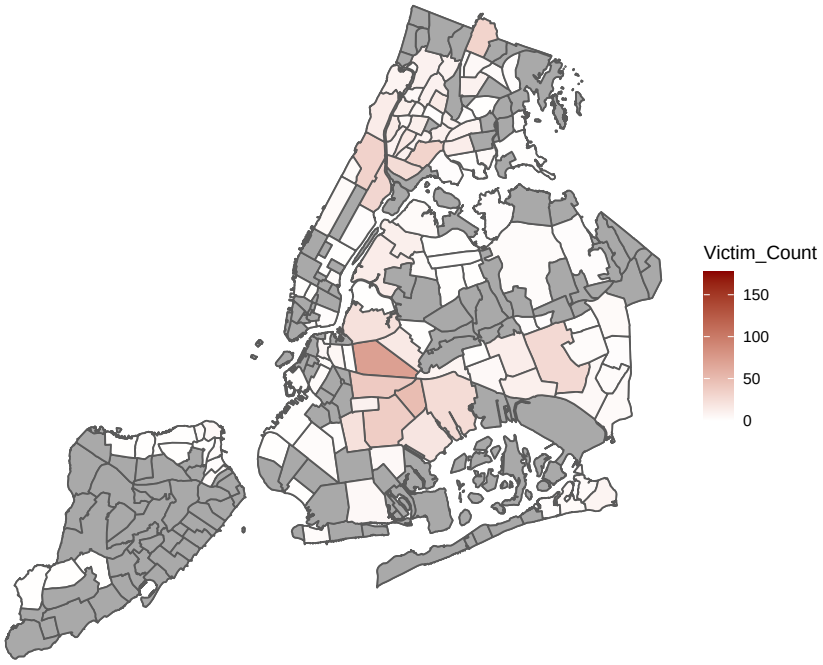
```
pivot_df <-  count_df_2 %>%
    pivot_wider(names_from=OCCUR_YR, values_from = n)

join_sf_2 <- nyc_neighborhoods_sf %>%
  left_join(pivot_df)

for (i in seq.int(last_yr,last_yr-numberOfYears,by=-1)) {

  column <- format(i)

  filter_sf_2 <- join_sf_2 %>%
    select(neighborhood,column) %>%
    rename(Victim_Count = column)

  plot_transform_2 <- ggplot() +
  geom_sf(data=nyc_neighborhoods_sf) +
  geom_sf(data=filter_sf_2, aes(fill=Victim_Count)) +
  theme_void() +
  scale_fill_gradient2(high="darkred",na.value = 'darkgray',limits=c(0,max_range)) +
  ggtitle(sprintf("NYC Shooting Victim Density by Neighborhood (%s)",column))

  print(plot_transform_2)
}
```



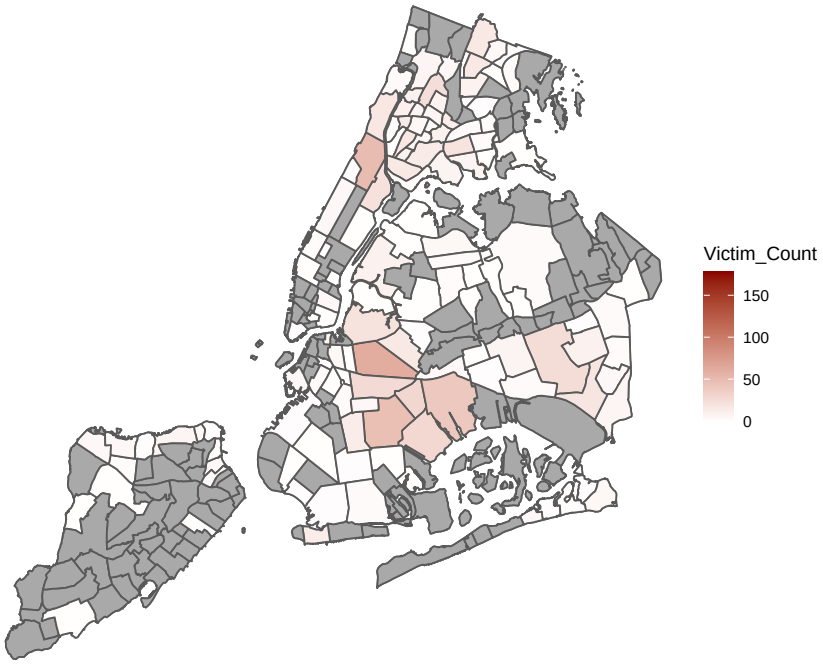
NYC Shooting Victim Density by Neighborhood (2019)



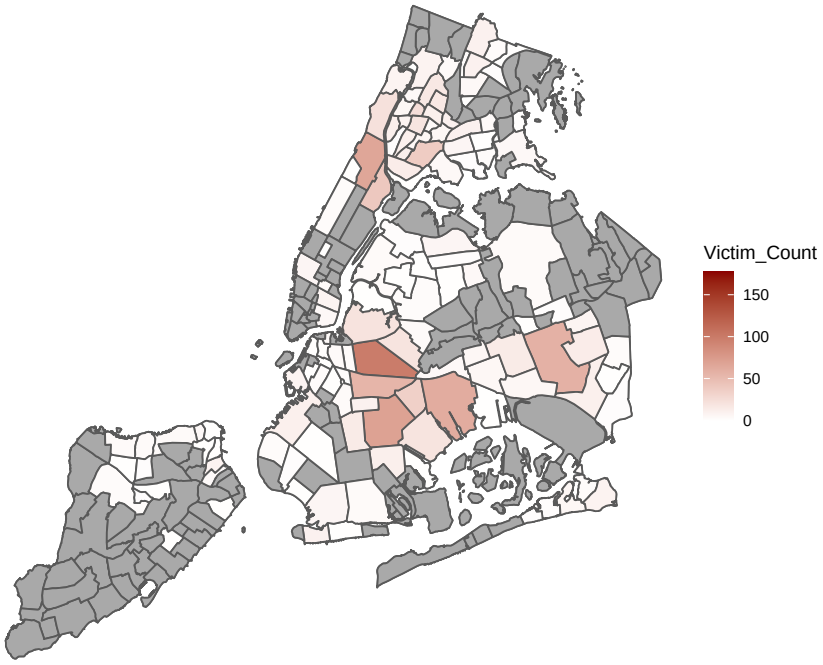
NYC Shooting Victim Density by Neighborhood (2018)



NYC Shooting Victim Density by Neighborhood (2017)



NYC Shooting Victim Density by Neighborhood (2016)



Geospatial Trend Conclusion

As you can see, the same neighborhoods show up with high victim rates and the total number of victims is variable each year. You'll want to keep in mind that we always need to question how these incidents are reported. The data footnotes explain that for a shooting to count as an incident, there must be a wounded

victim.

If you look closely at the color scale for each map, you'll notice that I have ensured they are consistent between each map so that we can compare both year over year and each neighborhood in the year.

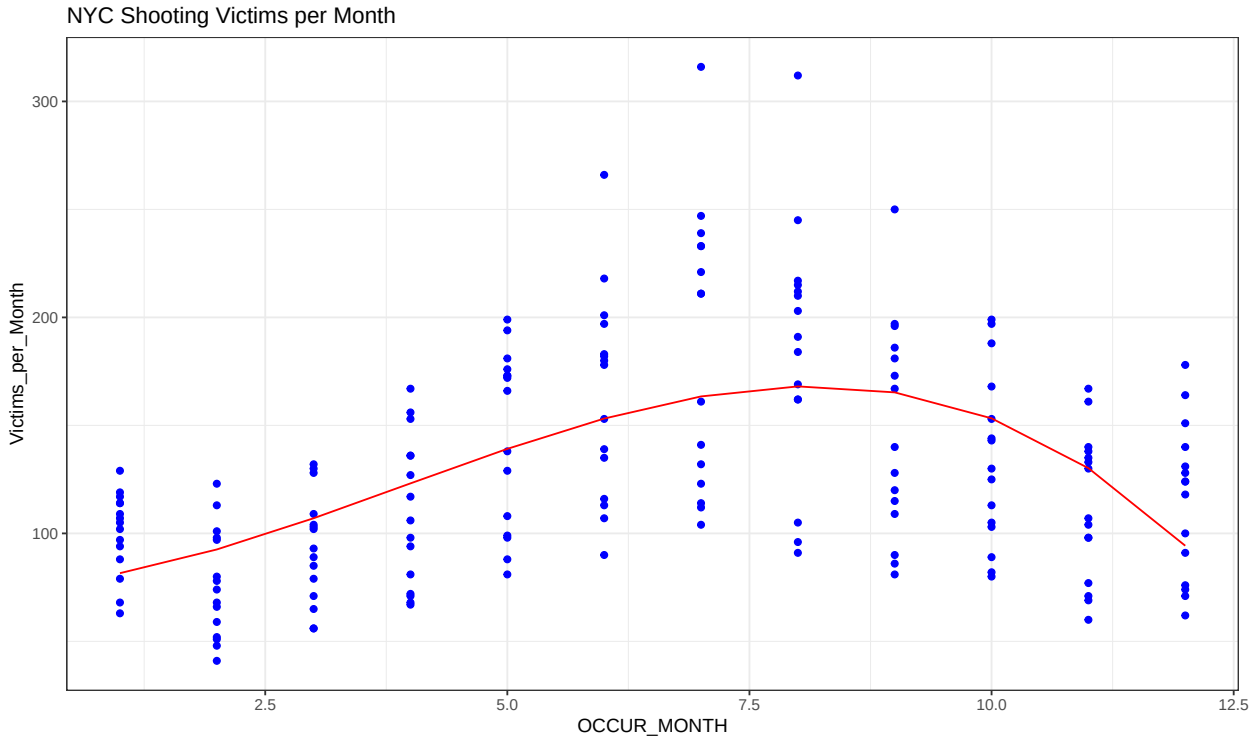
I encourage you to play around with it yourself and change the number of years if you can.

Modelling Seasonality

For the model, we will use the month as a proxy for season. Let's see if there is a relationship between the number of shootings victims and the month they occurred in. Again, this is not a statistics course, so we will not be doing any statistical tests. Much like the previous section, this will be a visual exercise only.

Below you'll see a graph with a point for each month at the total number of victims in that month.

```
model_data <- shooting_data %>%
  mutate(OCCUR_MONTH = month(OCCUR_DATE))  %>%
  select(OCCUR_YR,OCCUR_MONTH,INCIDENT_KEY)

month_count <- dplyr::count(model_data,OCCUR_YR,OCCUR_MONTH, name="Victims_per_Month")

model_data_gb_2 <- setNames(aggregate(month_count$Victims_per_Month,list(month_count$OCCUR_MONTH),FUN=m
  mutate(Mo2 = Month^2) %>%
  mutate(Mo3 = Month^3) %>%
  mutate(Mo4 = Month^4)

mod <- lm(Average ~ Month + Mo2 + Mo3, data = model_data_gb_2)

model_data_2 <- mutate(model_data_gb_2,pred=predict(mod))

ggplot() +
  geom_point(data=month_count,aes(x=OCCUR_MONTH, y=Victims_per_Month), color="blue") +
  geom_line(data=model_data_2,aes(x=Month, y=pred), color="red") +
  theme_bw() +
  ggtitle(sprintf("NYC Shooting Victims per Month"))
```



Model Conclusions

Here we can see two things. 1. There is likely some seasonality in the number of shooting victims in aggregate 2. We should summarize by incident as the volume of victims per incident could be skewing our results.

Bias

Bias in this data set could stem from the socioeconomic status of the particular neighborhoods as well as the sub-cultural and social institutions which may exist. These terms are very loaded and difficult to pinpoint. There is much we could do (assuming we have the data) to normalize our model in an effort to mitigate such bias. For example, if an area has 100 shootings in a month and another has 10, we should look to see if the first one is 10x larger in population than the second. Finally, referring back to #2 above, our results might be skewed if there are an out-sized number of victims per incident in any particular incident.

Final Thought

Other than highlighting what neighborhood has the highest victims, any deeper analysis into why this particular neighborhood is a hot spot (so to say) should account for population density, socioeconomic status, and/or land area. Shooting victims/incidents could be higher in lower income or gang neighborhoods and/or higher in high population areas. This topic is very complex and nuanced and any fair representation should be as thorough as possible.

Thanks for checking out my project!

```
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.5 (2021-03-31)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/Denver
##  date     2021-10-01
##
## - Packages -------------------------------------------------------------------
##  package      * version  date       lib source
##  abind          1.4-5    2016-07-21 [1] CRAN (R 4.0.2)
##  assertthat     0.2.1    2019-03-21 [1] CRAN (R 4.0.2)
##  backports      1.2.1    2020-12-09 [1] CRAN (R 4.0.2)
##  base64enc      0.1-3    2015-07-28 [1] CRAN (R 4.0.2)
##  bit            4.0.4    2020-08-04 [1] CRAN (R 4.0.2)
##  bit64          4.0.5    2020-08-30 [1] CRAN (R 4.0.2)
##  broom        * 0.7.9    2021-07-27 [1] CRAN (R 4.0.2)
##  cellranger     1.1.0    2016-07-27 [1] CRAN (R 4.0.2)
##  class          7.3-18   2021-01-24 [1] CRAN (R 4.0.5)
##  classInt       0.4-3    2020-04-07 [1] CRAN (R 4.0.2)
##  cli            3.0.1    2021-07-17 [1] CRAN (R 4.0.2)
##  codetools      0.2-18   2020-11-04 [1] CRAN (R 4.0.5)
##  colorspace     2.0-2    2021-06-24 [1] CRAN (R 4.0.2)
##  crayon         1.4.1    2021-02-08 [1] CRAN (R 4.0.2)
##  crosstalk      1.1.1    2021-01-12 [1] CRAN (R 4.0.2)
##  curl           4.3      2019-12-02 [1] CRAN (R 4.0.1)
##  DBI            1.1.1    2021-01-15 [1] CRAN (R 4.0.2)
##  dbplyr         2.1.1    2021-04-06 [1] CRAN (R 4.0.2)
##  dichromat      2.0-0    2013-01-24 [1] CRAN (R 4.0.2)
##  digest         0.6.27   2020-10-24 [1] CRAN (R 4.0.2)
##  dplyr        * 1.0.7    2021-06-18 [1] CRAN (R 4.0.2)
##  e1071          1.7-9    2021-09-16 [1] CRAN (R 4.0.2)
##  ellipsis       0.3.2    2021-04-29 [1] CRAN (R 4.0.2)
##  evaluate       0.14     2019-05-28 [1] CRAN (R 4.0.1)
##  fansi          0.4.2    2021-01-15 [1] CRAN (R 4.0.2)
##  farver         2.1.0    2021-02-28 [1] CRAN (R 4.0.2)
##  fastmap        1.1.0    2021-01-25 [1] CRAN (R 4.0.2)
##  forcats      * 0.5.1    2021-01-27 [1] CRAN (R 4.0.2)
##  foreign      * 0.8-81   2020-12-22 [1] CRAN (R 4.0.2)
##  fs             1.5.0    2020-07-31 [1] CRAN (R 4.0.2)
##  generics       0.1.0    2020-10-31 [1] CRAN (R 4.0.2)
##  ggplot2      * 3.3.5    2021-06-25 [1] CRAN (R 4.0.2)
##  glue           1.4.2    2020-08-27 [1] CRAN (R 4.0.2)
##  gtable         0.3.0    2019-03-25 [1] CRAN (R 4.0.2)
##  haven          2.4.3    2021-08-04 [1] CRAN (R 4.0.2)
##  highr          0.8      2019-03-20 [1] CRAN (R 4.0.2)
##  hms            1.1.0    2021-05-17 [1] CRAN (R 4.0.2)
```

```
##  htmltools      0.5.2    2021-08-25 [1] CRAN (R 4.0.2)
##  htmlwidgets    1.5.4    2021-09-08 [1] CRAN (R 4.0.2)
##  httr         * 1.4.2    2020-07-20 [1] CRAN (R 4.0.2)
##  jsonlite       1.7.2    2020-12-09 [1] CRAN (R 4.0.2)
##  KernSmooth     2.23-18  2020-10-29 [1] CRAN (R 4.0.5)
##  knitr          1.31     2021-01-27 [1] CRAN (R 4.0.2)
##  labeling       0.4.2    2020-10-20 [1] CRAN (R 4.0.2)
##  lattice        0.20-41  2020-04-02 [1] CRAN (R 4.0.5)
##  leafem         0.1.6    2021-05-24 [1] CRAN (R 4.0.2)
##  leaflet        2.0.4.1  2021-01-07 [1] CRAN (R 4.0.2)
##  leafsync       0.1.0    2019-03-05 [1] CRAN (R 4.0.2)
##  lifecycle      1.0.0    2021-02-15 [1] CRAN (R 4.0.2)
##  lubridate    * 1.7.10   2021-02-26 [1] CRAN (R 4.0.2)
##  lwgeom         0.2-7    2021-07-28 [1] CRAN (R 4.0.2)
##  magrittr       2.0.1    2020-11-17 [1] CRAN (R 4.0.2)
##  maps         * 3.3.0    2018-04-03 [1] CRAN (R 4.0.2)
##  maptools       1.1-2    2021-09-07 [1] CRAN (R 4.0.2)
##  modelr         0.1.8    2020-05-19 [1] CRAN (R 4.0.2)
##  munsell        0.5.0    2018-06-12 [1] CRAN (R 4.0.2)
##  pillar         1.6.2    2021-07-29 [1] CRAN (R 4.0.2)
##  pkgconfig      2.0.3    2019-09-22 [1] CRAN (R 4.0.2)
##  png            0.1-7    2013-12-03 [1] CRAN (R 4.0.2)
##  proxy          0.4-26   2021-06-07 [1] CRAN (R 4.0.2)
##  purrr        * 0.3.4    2020-04-17 [1] CRAN (R 4.0.2)
##  R6             2.5.0    2020-10-28 [1] CRAN (R 4.0.2)
##  rappdirs       0.3.3    2021-01-31 [1] CRAN (R 4.0.2)
##  raster         3.4-13   2021-06-18 [1] CRAN (R 4.0.2)
##  RColorBrewer   1.1-2    2014-12-07 [1] CRAN (R 4.0.2)
##  Rcpp         * 1.0.7    2021-07-07 [1] CRAN (R 4.0.2)
##  readr        * 2.0.1    2021-08-10 [1] CRAN (R 4.0.2)
##  readxl         1.3.1    2019-03-13 [1] CRAN (R 4.0.2)
##  reprex         2.0.1    2021-08-05 [1] CRAN (R 4.0.2)
##  rgdal        * 1.5-27   2021-09-16 [1] CRAN (R 4.0.2)
##  rgeos        * 0.5-7    2021-09-07 [1] CRAN (R 4.0.2)
##  rlang          0.4.11   2021-04-30 [1] CRAN (R 4.0.2)
##  rmarkdown      2.11     2021-09-14 [1] CRAN (R 4.0.5)
##  rstudioapi     0.13     2020-11-12 [1] CRAN (R 4.0.2)
##  rvest          1.0.1    2021-07-26 [1] CRAN (R 4.0.2)
##  s2             1.0.6    2021-06-17 [1] CRAN (R 4.0.2)
##  scales       * 1.1.1    2020-05-11 [1] CRAN (R 4.0.2)
##  sessioninfo    1.1.1    2018-11-05 [1] CRAN (R 4.0.2)
##  sf           * 1.0-2    2021-07-26 [1] CRAN (R 4.0.2)
##  sp           * 1.4-5    2021-01-10 [1] CRAN (R 4.0.2)
##  stars          0.5-3    2021-06-08 [1] CRAN (R 4.0.2)
##  stringi        1.5.3    2020-09-09 [1] CRAN (R 4.0.2)
##  stringr      * 1.4.0    2019-02-10 [1] CRAN (R 4.0.2)
##  tibble       * 3.1.0    2021-02-25 [1] CRAN (R 4.0.2)
##  tidyr        * 1.1.3    2021-03-03 [1] CRAN (R 4.0.2)
##  tidyselect   * 1.1.1    2021-04-30 [1] CRAN (R 4.0.2)
##  tidyverse    * 1.3.1    2021-04-15 [1] CRAN (R 4.0.2)
##  tigris       * 1.4.1    2021-06-18 [1] CRAN (R 4.0.2)
##  tmap         * 3.3-2    2021-06-16 [1] CRAN (R 4.0.2)
##  tmaptools      3.1-1    2021-01-19 [1] CRAN (R 4.0.2)
##  tzdb           0.1.2    2021-07-20 [1] CRAN (R 4.0.2)
```

```
##  units          0.7-2    2021-06-08 [1] CRAN (R 4.0.2)
##  utf8           1.2.1    2021-03-12 [1] CRAN (R 4.0.2)
##  uuid           0.1-4    2020-02-26 [1] CRAN (R 4.0.2)
##  vctrs          0.3.8    2021-04-29 [1] CRAN (R 4.0.2)
##  viridisLite    0.4.0    2021-04-13 [1] CRAN (R 4.0.2)
##  vroom          1.5.5    2021-09-14 [1] CRAN (R 4.0.5)
##  withr          2.4.1    2021-01-26 [1] CRAN (R 4.0.2)
##  wk             0.5.0    2021-07-13 [1] CRAN (R 4.0.2)
##  xfun           0.26     2021-09-14 [1] CRAN (R 4.0.5)
##  XML            3.99-0.8 2021-09-17 [1] CRAN (R 4.0.2)
##  xml2           1.3.2    2020-04-23 [1] CRAN (R 4.0.2)
##  yaml           2.2.1    2020-02-01 [1] CRAN (R 4.0.2)
##
## [1] /Library/Frameworks/R.framework/Versions/4.0/Resources/library
```