Little Free Library Analysis

Kaleb Crans

2023-04-17

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(spData)
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
rm(list = ls())
```

```
if (file.exists("lfl.RData")) {
  load("lfl.RData")
} else {
  libraries = as_tibble(read_csv("libraries.csv"))
  save(libraries, file = "lfl.RData")
}
```

Data Cleaning and Preparation

Let's take a look at the different data types of the columns:

```
lapply(libraries, typeof)
```

```
## $id
## [1] "double"
##
## $Name
## [1] "character"
##
## $Street__c
## [1] "character"
##
```

```
## $City__c
## [1] "character"
##
## $State_Province_Region__c
## [1] "character"
##
## $Postal_Zip_Code__c
## [1] "character"
##
## $Country__c
## [1] "character"
##
## $Traveling_Library__c
## [1] "logical"
##
## $Official_Charter_Number__c
## [1] "character"
##
## $First_Map_Date__c
## [1] "double"
##
## $Map_Me__c
## [1] "character"
##
## $Map_Date__c
## [1] "double"
##
## $Duplicate_Charter_Number__c
## [1] "logical"
##
## $Count_of_Primary_Stewards__c
## [1] "double"
##
## $Latitude_MapAnything__c
## [1] "double"
##
## $Longitude_MapAnything__c
## [1] "double"
##
## $Library_Geolocation__Latitude__s
## [1] "double"
##
## $Library_Geolocation__Longitude__s
## [1] "double"
##
## $check_in_count
## [1] "double"
```

Map_Me__c needs to be transformed into a logical variable, as "Taken Down Temporarily" and "Mapped" are the only two categories.

```
libraries <- libraries %>% mutate(Map_Me__c = Map_Me__c ==  "Mapped")
```

Note that there is only one row with this flag set to false:

```
sum(libraries$Map_Me__c == FALSE)
```

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

A quick look at check_in_counts shows that this feature is rarely used when you consider how many times a given library is actually visited:

```
print("Max:")
```

```
## [1] "Max:"
```

```
max(libraries$check_in_count)
```

```
## [1] 47
```

```
print("Summary Stats:")
```

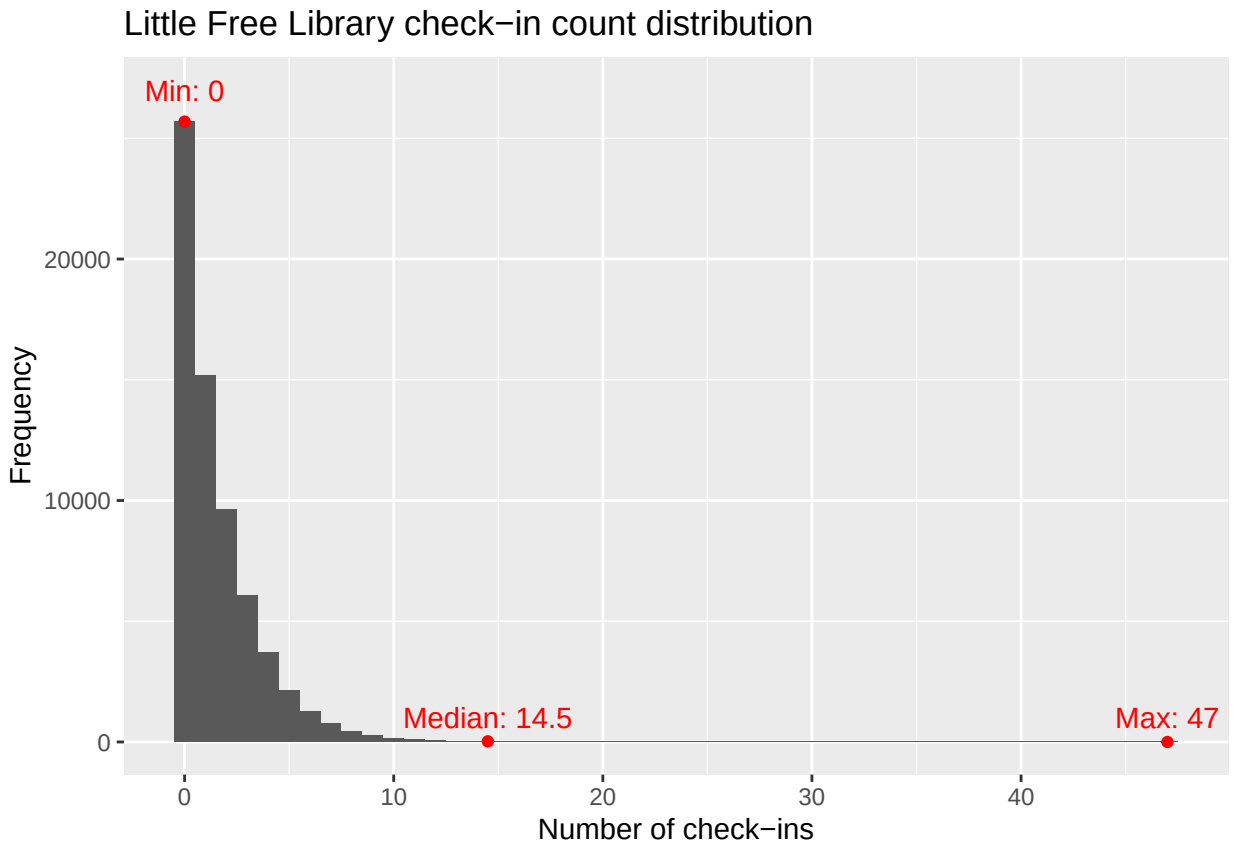
```
## [1] "Summary Stats:"
```

```
summary(libraries$check_in_count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.567   2.000  47.000
```

```
check_in_dist <- libraries %>% count(check_in_count)

annotations <-  tibble(x = c(min(check_in_dist$check_in_count), median(check_in_dist$check_in_count), ma
                       y = c(27000, 1000, 1000),
                       label = c("Min:", "Median:", "Max:"))
ggplot(data = libraries, aes(x =  check_in_count)) +
  ggtitle("Little Free Library check-in count distribution") +
  xlab("Number of check-ins") +
  ylab("Frequency") +
  geom_histogram(binwidth = 1) +
  geom_point(data = tibble(x = annotations$x, y = c(25691, mean(c(41, 17)), 1)), aes(x = x, y = y), col
  geom_text(data = annotations, aes(x = x, y = y, label = paste(label, x)), size = 4, color = "red")
```




```
  #ggsave("checkins.png")
```

```
# Clean up alternative names for the same country:
libraries <- libraries %>% mutate(Country__c = replace(Country__c, Country__c %in% c("USA", "US", "U.S.

# Ranking of countries by number of little free libraries:
libraries %>% count(Country__c) %>% arrange(desc(n)) %>% head(10)
```

```
##        Country__c     n
## 1   United States 59816
## 2          Canada  2871
## 3            <NA>   633
## 4           Italy   367
## 5  United Kingdom   278
## 6       Australia   229
## 7          France   202
## 8     Philippines   123
## 9     Netherlands   108
## 10        Belgium    90
```

The U.S. by and large has the greatest amount of little free libraries (with alternate spellings outpacing many countries even). Canada is the only country with a somewhat comparable amount, specifically if you adjust for population size.

```
# US population
us_pop <- 331900000

# Canada population
can_pop <- 38250000
```

```
us_count <- libraries %>% filter(Country__c == "United States") %>% nrow

can_count <- libraries %>% filter(Country__c == "Canada") %>% nrow
```

So the per capita number of little free libraries in the US is:

```
format(us_count/us_pop, scientific = FALSE)
```

```
## [1] "0.000180223"
```

And in Canada is:

```
format(can_count/can_pop, scientific = FALSE)
```

```
## [1] "0.00007505882"
```

So we can conclude that Little Free Libraries are a predominately American phenomenon. For the purposes of further analysis, let's exclude all data points not in the US. We will also include rows that have a NA country value (but still a state code) as on inspection many of them are actually located in the US. We can impute the correct country code based off the coordinates later.

```
libraries <- libraries %>% filter(Country__c == "United States" | (is.na(Country__c) & !is.na(State_Pro
```

Analysis by state

How about the distribution by state?

```
length(unique(libraries$State_Province_Region__c))
```

```
## [1] 174
```

But there's only 50 states! So we need to do some data cleaning first.

```
libraries %>% count(State_Province_Region__c) %>% arrange(desc(n)) %>% tail(20)
```

```
##     State_Province_Region__c n
## 155                      USA 1
## 156                       Ut 1
## 157                  Wyoming 1
## 158                        X 1
## 159                       co 1
## 160                       ct 1
## 161                 kentucky 1
## 162                       ks 1
## 163                       ma 1
## 164                       md 1
## 165                       mi 1
## 166                       mo 1
## 167                       ny 1
## 168                     ohio 1
## 169                       ok 1
## 170                    omaha 1
## 171                       pa 1
## 172                       tn 1
## 173                       va 1
## 174                       wv 1
```

There's a bunch of different spelling variations. Let's instead take the actual coordinates and then find the states ourselves (assuming the coordinates are correct). One point of interest in the dataset to note is that

there are two sets of coordinates for each row: Latitude__MapAnything___c and Longitude__MapAnything___c vs Library__Geolocation___Latitude___s and Library__Geolocation___Longitude___s.

We can make a dataframe with the differences as separate columns, and print out the mean difference in latitude and longitude respectively:

```
differences <- libraries %>% mutate(dif_lat = (abs(Latitude_MapAnything__c) - abs(Library_Geolocation__l
c(mean(differences$dif_lat), mean(differences$dif_long))
```

```
## [1] -1.502292 -3.769037
```

Unfortunately due to the curvature of the earth these values as-is don't reflect physical distances

To explore further, let's take one example with a latitude difference of 16.6 and then plug the coordinates into Google maps. We get two different locations, one in Lake Park Iowa and the other in Lake Park Florida:

```
libraries %>% filter(Latitude_MapAnything__c == 26.79489)
```

```
##      id          Name   Street__c   City__c State_Province_Region__c
## 1 41210 LIB-000004180 307 4th St. Lake Park                       FL
##   Postal_Zip_Code__c    Country__c Traveling_Library__c
## 1              33403 United States               FALSE
##   Official_Charter_Number__c First_Map_Date__c Map_Me__c Map_Date__c
## 1                      11455        2014-10-03      TRUE  2019-01-22
##   Duplicate_Charter_Number__c Count_of_Primary_Stewards__c
## 1                       FALSE                            1
##   Latitude_MapAnything__c Longitude_MapAnything__c
## 1                26.79489               -80.06038
##   Library_Geolocation__Latitude__s Library_Geolocation__Longitude__s
## 1                         43.45611                         -95.31709
##   check_in_count
## 1              0
```

The MapAnything location:

The geolocation:

This library is actually displayed incorrectly in Iowa on the official webapp. The correct coordinates for this address are in Florida.

Another example is a location with a 101 degree difference in longitude.

```
libraries %>% filter(Latitude_MapAnything__c == 37.33889)
```

```
##      id          Name                                          Street__c
## 1 75881 LIB-000084125 North Campus Building, Perandori Dushan, Mitrovicë
##     City__c State_Province_Region__c Postal_Zip_Code__c    Country__c
## 1 Mitrovica                      KS              40000 United States
##   Traveling_Library__c Official_Charter_Number__c First_Map_Date__c Map_Me__c
## 1                FALSE                      85849        2022-10-13      TRUE
##   Map_Date__c Duplicate_Charter_Number__c Count_of_Primary_Stewards__c
## 1  2022-10-13                      FALSE                            1
##   Latitude_MapAnything__c Longitude_MapAnything__c
## 1                37.33889               -121.8825
##   Library_Geolocation__Latitude__s Library_Geolocation__Longitude__s
## 1                         42.89542                          20.86808
##   check_in_count
## 1              0
```

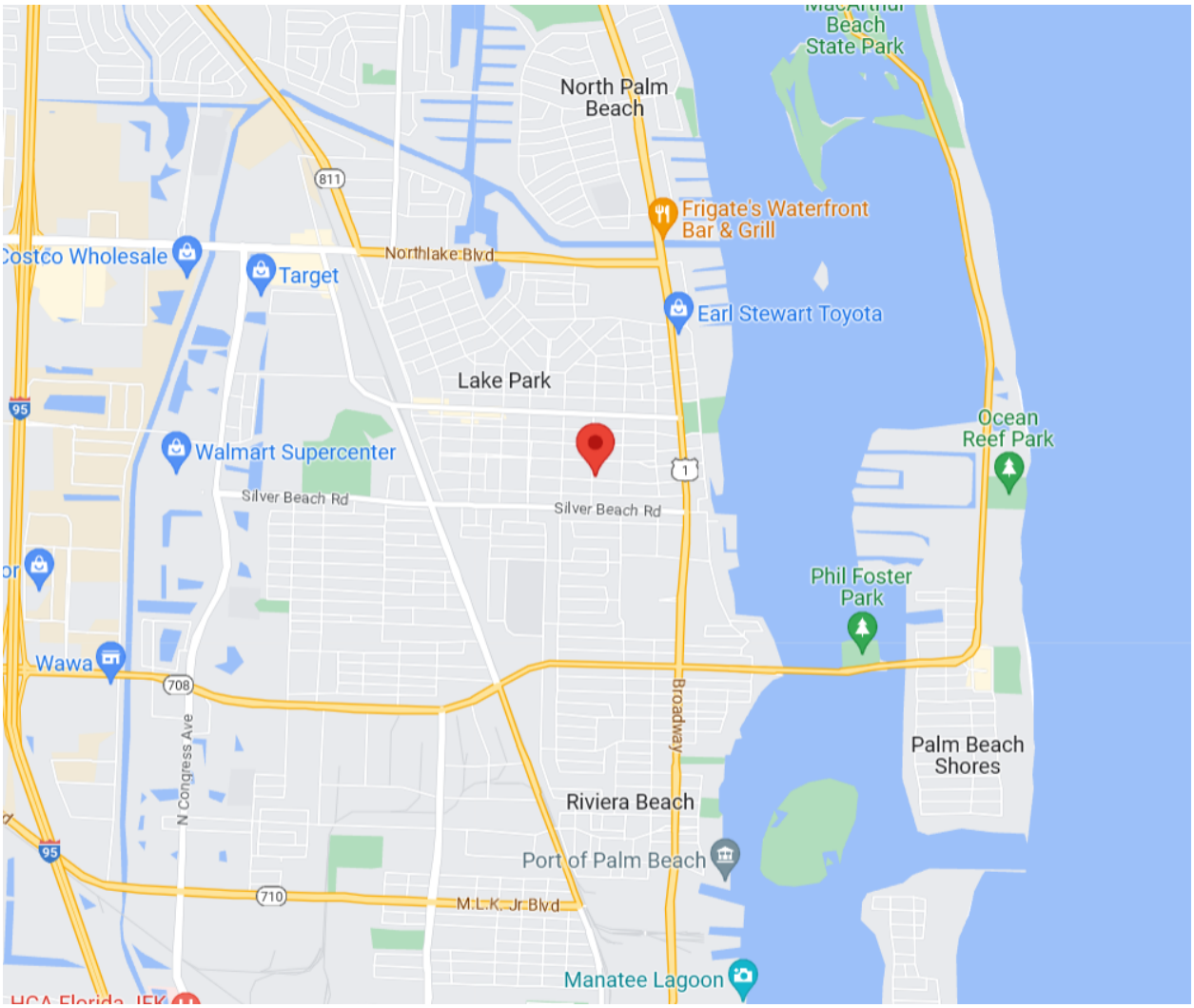


Figure 1: Lake Park in Florida

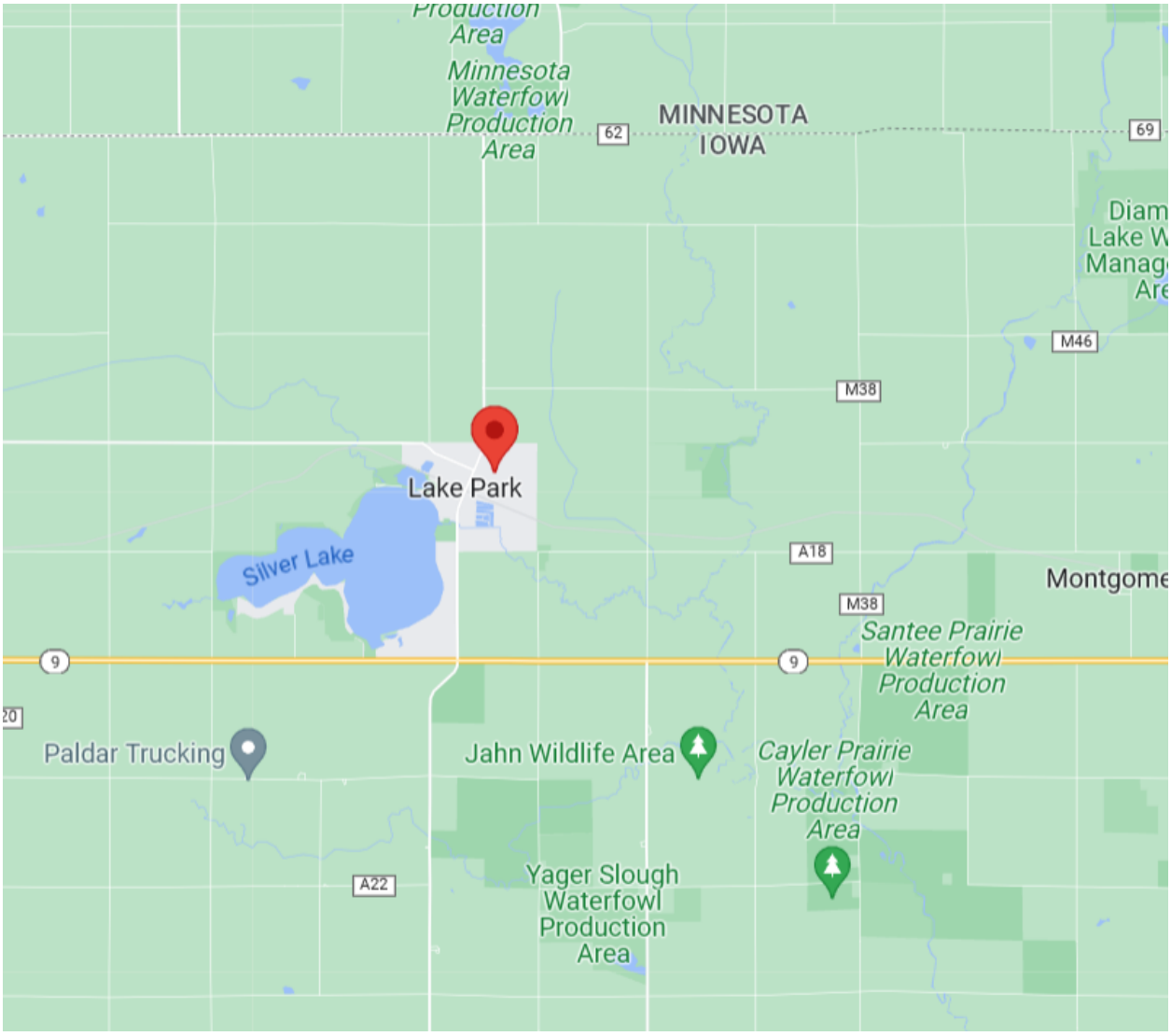


Figure 2: Lake Park in Iowa

The actual location is in Kosovo, but because they put "KS" as the state (which is Kansas, not Kosovo) this row was mistakenly assigned "United States" as its country.

The MapAnything location is in San Jose:

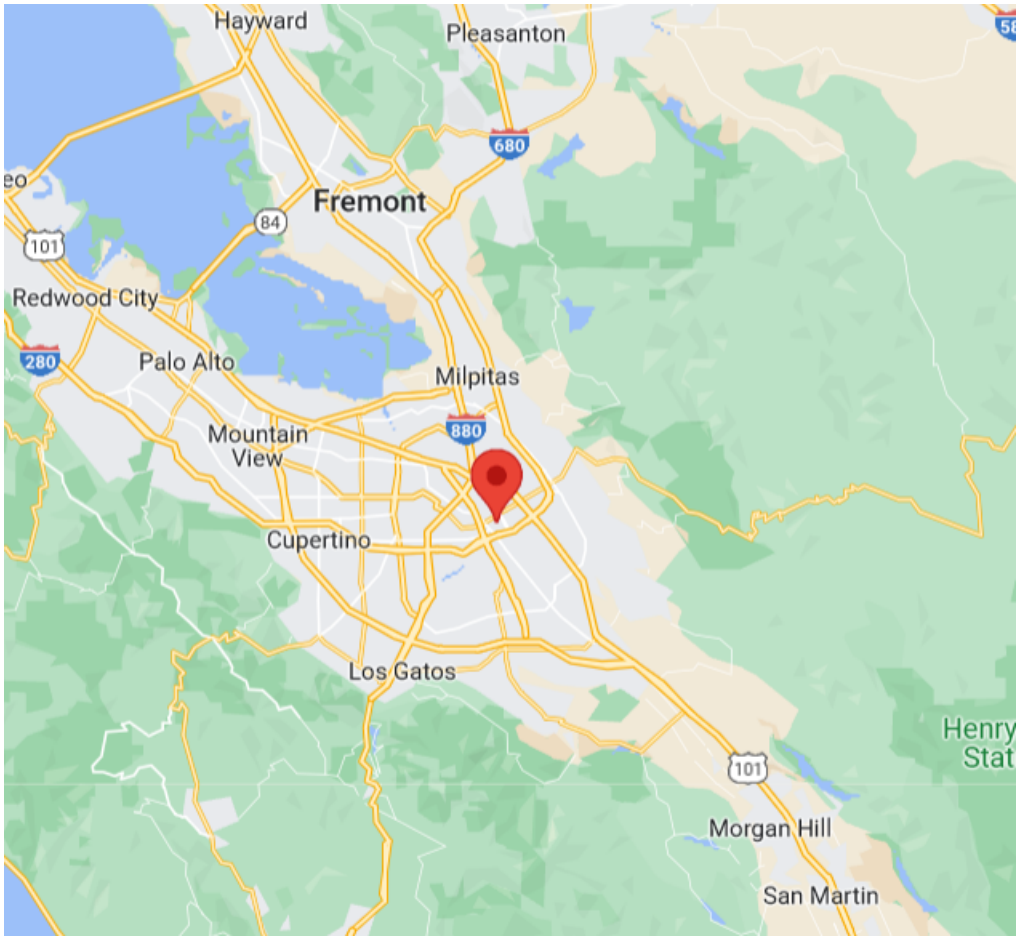


Figure 3: Location in San Jose

and the Geolocation is in Kosovo:

So we have two different examples where the correct coordinates are of different types. If we look at the distribution of coordinates we have:

```
libraries %>% select(Latitude_MapAnything__c, Library_Geolocation__Latitude__s, Longitude_MapAnything__
```

```
##  Latitude_MapAnything__c Library_Geolocation__Latitude__s
##  Min.   :-26.14          Min.   :-27.00
##  1st Qu.: 34.43          1st Qu.: 35.17
##  Median : 39.58          Median : 39.74
##  Mean   : 37.51          Mean   : 39.01
##  3rd Qu.: 42.44          3rd Qu.: 42.50
##  Max.   : 71.30          Max.   : 86.95
##  Longitude_MapAnything__c Library_Geolocation__Longitude__s
##  Min.   :-170.47          Min.   :-170.49
##  1st Qu.:-105.06          1st Qu.:-105.96
##  Median : -88.09          Median : -88.64
##  Mean   : -89.96          Mean   : -93.56
```

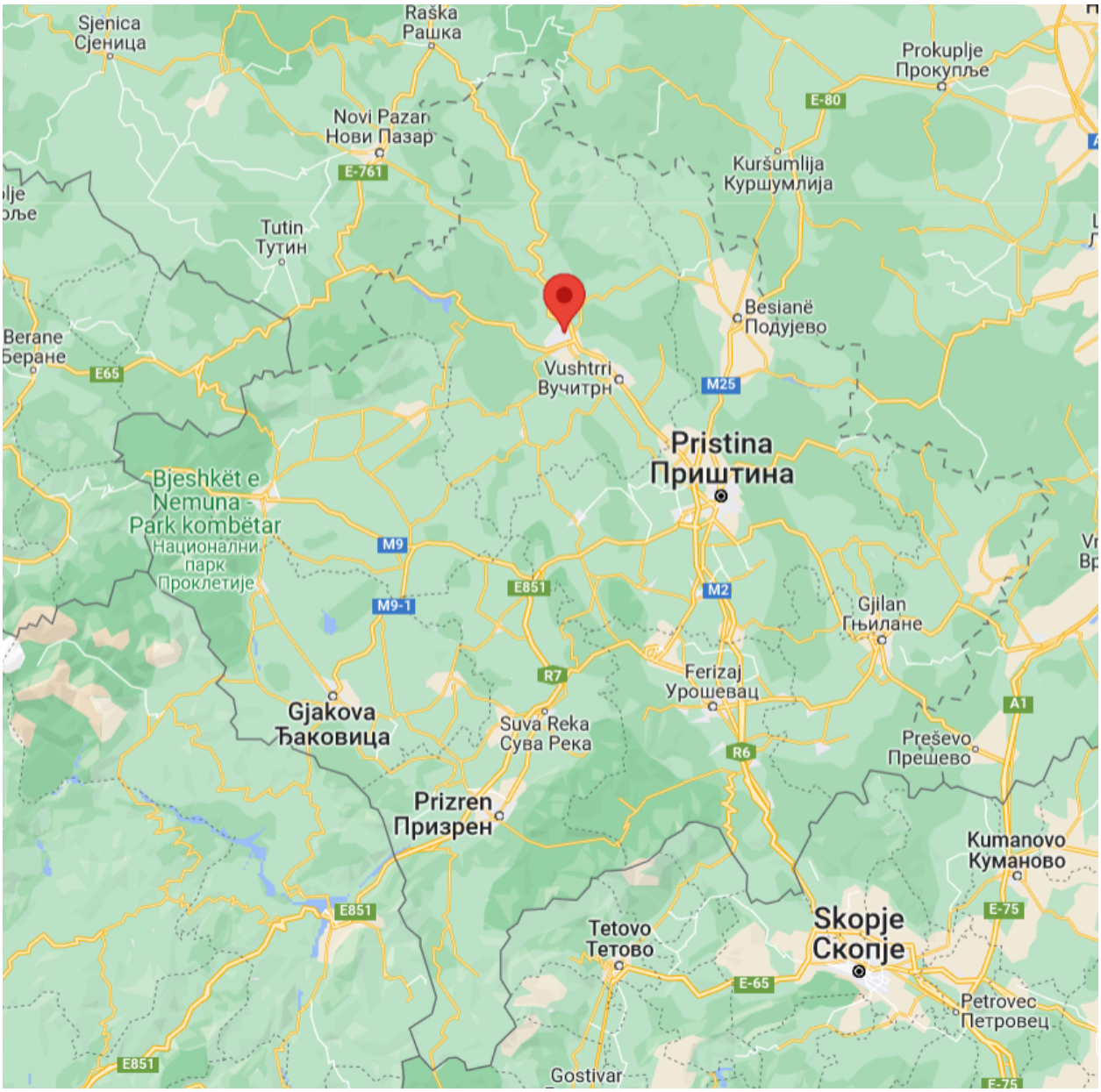


Figure 4: Location in Kosovo

```
##  3rd Qu.: -79.91          3rd Qu.: -80.82
##  Max.   :  77.56          Max.   : 158.07
```

Thus summary statistics are similar, but there are enough differences to cause concern. Note that there are a decent amount of rows where the MapAnything coordinates are (0, 0):

```
libraries %>% filter(Latitude_MapAnything__c == 0 & Longitude_MapAnything__c == 0) %>% count()
```

```
##      n
## 1 2428
```

One example is the library with id 14180.

```
libraries %>% filter(id == 14180)
```

```
##      id          Name      Street__c    City__c State_Province_Region__c
## 1 14180 LIB-000038331 11509 Kenny Dr Fort Worth                        TX
##   Postal_Zip_Code__c    Country__c Traveling_Library__c
## 1              76244 United States               FALSE
##   Official_Charter_Number__c First_Map_Date__c Map_Me__c Map_Date__c
## 1                      77029        2019-03-21      TRUE  2023-04-07
##   Duplicate_Charter_Number__c Count_of_Primary_Stewards__c
## 1                       FALSE                            1
##   Latitude_MapAnything__c Longitude_MapAnything__c
## 1                       0                       0
##   Library_Geolocation__Latitude__s Library_Geolocation__Longitude__s
## 1                         32.93978                         -97.27757
##   check_in_count
## 1              1
```

None of the values look notable other than the (0, 0) MapAnything coordinates, and this library shows up on the official map.

These (0, 0) coordinates are basically missing values as all the libraries we are looking at are located in the US so (0, 0) is definitely an invalid coordinate. If we look at the webapp, it appears that the developers use the geolocation values on the interactive map:

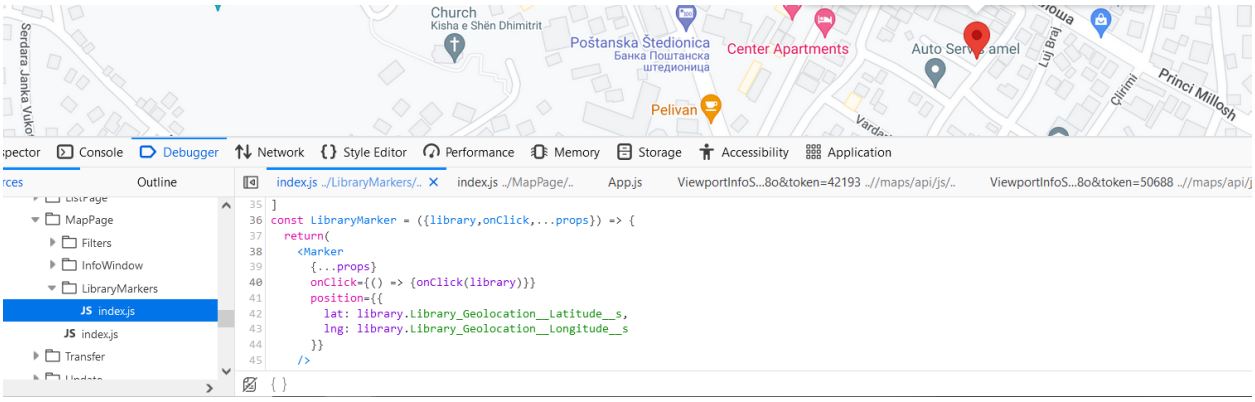


Figure 5: Screenshot of javascript snippet using Library_Geolocation for the pins on the map

So let's default to the geolocation coordinates, but use the MapAnything coordinates if they match up with the address listed. Let's then drop every data point with coordinates not located within the United States. In order to determine this we can look at what U.S. state a given library is in and then filter out the libraries with no state values.

```
# Convert the coordinates to a sf object
# Our coordinate reference system is the WGS84 standard which is what Google
# maps uses. Its EPSG Code is 4326. The format for a point is (longitude, latitude).
lib_pts <- libraries %>% st_as_sf(coords = c("Library_Geolocation__Longitude__s", "Library_Geolocation_

# For comparison also convert the MapAnything coordinates into a sf object
lib_pts_alt <- libraries %>% st_as_sf(coords = c("Longitude_MapAnything__c", "Latitude_MapAnything__c")

# Read in and transform the GADM data to WGS84 format.
GADM_data <- st_read(dsn = "gadm36_USA_gpkg/gadm36_USA.gpkg", layer = "gadm36_USA_1")
```

```
## Reading layer `gadm36_USA_1' from data source
##   `/Users/kcrans/Desktop/projects/little_free/gadm36_USA_gpkg/gadm36_USA.gpkg'
##   using driver `GPKG'
## Simple feature collection with 51 features and 10 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -179.1506 ymin: 18.90986 xmax: 179.7734 ymax: 72.6875
## Geodetic CRS:  WGS 84
```

```
state_pts <- st_transform(GADM_data, crs = 4326)

# Make a data.frame of all the possible state names
state_names <- state_pts$NAME_1

# Find the intersections between the library points and state polygons
# Convert to an integer to use as an index in the state names data.frame.
classifications <- as.integer(st_intersects(lib_pts, state_pts))
alt_classifications <- as.integer(st_intersects(lib_pts_alt, state_pts))

# Store the results in a copy of libraries
temp_libraries <- libraries %>% mutate(
  state = state_names[classifications], alt_state = state_names[alt_classifications])
sum(is.na(temp_libraries$state) & is.na(temp_libraries$alt_state))
```

```
## [1] 70
```

There are 70 locations with (both) coordinates not in the U.S. for whatever reason. Let's take a look at a few of them.

```
temp_libraries %>% filter(is.na(state) & is.na(alt_state)) %>% head()
```

```
##     id          Name                    Street__c           City__c
## 1 2278 LIB-000026204                   Perumkulam        Kottarakara
## 2 3919 LIB-000027852                Keerkring 112           De Meern
## 3 4141 LIB-000028075                   Aljazar st           Khartoum
## 4 4515 LIB-000028452 Chequers Corner, Hurst Drive Walton on the Hill
## 5 5604 LIB-000028915              1 Stirrup Close            Newbury
## 6 7355 LIB-000030684               Dorfstrasse 21             Bühler
##   State_Province_Region__c Postal_Zip_Code__c    Country__c
## 1                     <NA>             691566 United States
## 2                     <NA>             3454kz United States
## 3                     <NA>              11112 United States
## 4                     <NA>           KT20 7QT United States
## 5                     <NA>           RG14 7XD United States
## 6                     <NA>               9055 United States
```

```
##   Traveling_Library__c Official_Charter_Number__c First_Map_Date__c Map_Me__c
## 1                FALSE                      57110        2017-06-28      TRUE
## 2                FALSE                      58388        2017-08-29      TRUE
## 3                FALSE                      46879        2017-09-07      TRUE
## 4                FALSE                      46255        2017-09-28      TRUE
## 5                FALSE                      60235        2017-10-16      TRUE
## 6                FALSE                     M68171        2018-01-29      TRUE
##   Map_Date__c Duplicate_Charter_Number__c Count_of_Primary_Stewards__c
## 1  2017-06-28                      FALSE                            1
## 2  2017-08-29                      FALSE                            1
## 3  2017-09-07                      FALSE                            1
## 4  2017-09-28                      FALSE                            1
## 5  2017-10-16                      FALSE                            1
## 6  2018-02-19                      FALSE                            2
##   Latitude_MapAnything__c Longitude_MapAnything__c
## 1                 9.03882               76.7597700
## 2                52.08682                5.0297300
## 3                15.50065               32.5598994
## 4                51.27481               -0.2506241
## 5                51.38518               -1.3159300
## 6                47.37292                9.4252400
##   Library_Geolocation__Latitude__s Library_Geolocation__Longitude__s
## 1                          9.03882                         76.7597700
## 2                         52.08682                          5.0297300
## 3                         15.50065                         32.5598994
## 4                         51.27481                         -0.2506241
## 5                         51.38518                         -1.3159300
## 6                         47.37292                          9.4252400
##   check_in_count state alt_state
## 1              0  <NA>      <NA>
## 2              0  <NA>      <NA>
## 3              0  <NA>      <NA>
## 4              0  <NA>      <NA>
## 5              0  <NA>      <NA>
## 6              0  <NA>      <NA>
```

Some of the rows are for foreign libraries, but it look like the majority are libraries with no street entries. Some like charter number G10014(148 Marina Plaza Dunedin) are located very close to the ocean and hence were classified incorrectly due to the resolution of the geography in the GADM dataset. A few like 150219 (1710 S Trenon Ave Tulsa) are mislabeled with coordinates not in the Unites States. By manual inspection, it looks the MapAnything coordinates give the appropriate state for some of the cases where the locations are right near a body of water or the street address is missing. When a library is located in a country other than the United States, both columns will have NA values. In general, if we take a look at the state assignments based off the geolocation coordinates ("state") and MapAnything coordinates ("alt_state"), the possibilities are:

1. NA for both. This means either the library is not in the U.S and should be removed from our dataframe, or it is located in the U.S. but too close to the ocean. We filter out data in the former instance and use the "State_Province_Region___c" assignment in the latter.
2. An actual state for the geolocation coordinates and NA for MapAnything. We should use the "state" assignment then.
3. An actual state for the MapAnything coordinates and NA for the geolocation. We will go with the "alt_state" assignment in this case.

4. and 5. Actual states for both coordinates. If they are the same, we will go with that assignment. If they are different, we will use whichever assignment lines up with the "State_Province_Region___c" value for that datapoint.

Now, let's create a new column state_name to hold whatever state name based off the above criteria we choose, and also long and lat columns to hold the coordinates we end up using.

First, here's a helper function to convert state abbreviations into full names:

```
convert_state <- function(state_code) {
  up_code <- toupper(state_code) # Ignore case differences
  name_str <- switch(up_code,
  "AL" = "Alabama",
  "AK" = "Alaska",
  "AZ" = "Alaska",
  "AR" = "Arkansas",
  "AS" = "American Samoa",
  "CA" = "California",
  "CO" = "Colorado",
  "CT" = "Connecticut",
  "DE" = "Delaware",
  "DC" = "District of Columbia",
  "FL" = "Florida",
  "GA" = "Georgia",
  "GU" = "Guam",
  "HI" = "Hawaii",
  "ID" = "Idaho",
  "IL" = "Illinois",
  "IN" = "Indiana",
  "IA" = "Iowa",
  "KS" = "Kansas",
  "KY" = "Kentucky",
  "LA" = "Louisiana",
  "ME" = "Maine",
  "MD" = "Maryland",
  "MA" = "Massachusetts",
  "MI" = "Michigan",
  "MN" = "Minnesota",
  "MS" = "Mississippi",
  "MO" = "Missouri",
  "MT" = "Montana",
  "NE" = "Nebraska",
  "NV" = "Nevada",
  "NH" = "New Hampshire",
  "NJ" = "New Jersey",
  "NM" = "New Mexico",
  "NY" = "New York",
  "NC" = "North Carolina",
  "ND" = "North Dakota",
  "MP" = "Northern Mariana Islands",
  "OH" = "Ohio",
  "OK" = "Oklahoma",
  "OR" = "Oregon",
  "PA" = "Pennsylvania",
  "PR" = "Puerto Rico",
```

```
  "RI" = "Rhode Island",
  "SC" = "South Carolina",
  "SD" = "South Dakota",
  "TN" = "Tennessee",
  "TX" = "Texas",
  "TT" = "Trust Territories",
  "UT" = "Utah",
  "VT" = "Vermont",
  "VA" = "Virginia",
  "VI" = "Virgin Islands",
  "WA" = "Washington",
  "WV" = "West Virginia",
  "WI" = "Wisconsin",
  "WY" = "Wyoming",
  state_code # Else return the string as-is
  )
  return(name_str)
}
convert_state <- Vectorize(convert_state)
```

Let's build our dataframe case-by-case:

1.

```
temp_libraries %>% filter(is.na(state) & is.na(alt_state)) %>% filter(!is.na(State_Province_Region__c)
```

```
##       id          Name               Street__c             City__c
## 1  10894 LIB-000034171 1932 S. Oceanshore Blvd       Flagler Beach
## 2  19119 LIB-000046571   319 Dunham Point Road           Deer Isle
## 3  19356 LIB-000043089                    <NA>              Alytus
## 4  26622 LIB-000053055           1 Newport Ave     North Kingstown
## 5  34172 LIB-000064463       567 Angell Street          Providence
## 6  46526 LIB-000009684     510 Shorewood Drive International Falls
## 7  51495 LIB-000014854       86 Butts Rock Rd.      Little Compton
## 8  51556 LIB-000014915     Ocean Ave & Lincoln     Avon by the Sea
## 9  56156 LIB-000019673      2013 Wildwood Lane           Anchorage
## 10 58958 LIB-000022491          1800 gulf road      Tarpon springs
## 11 63301 LIB-000071612          185 Ferry Road        Saunderstown
## 12 64910 LIB-000073170         261 Hart Street             Dighton
## 13 75691 LIB-000083936      22 Beachwood Drive             Warwick
## 14 78630 LIB-000086854                    <NA>          Sandbridge
## 15 78786 LIB-000087009                    <NA>             Madison
## 16 79239 LIB-000087459   114 W. Chicago Street            Caldwell
## 17 82395 LIB-000090612       64 Sleeper Street              Boston
##    State_Province_Region__c Postal_Zip_Code__c    Country__c
## 1                        FL              32136 United States
## 2                        ME              04627 United States
## 3                        AL              63210 United States
## 4                        CT              02852 United States
## 5                        RI              02906 United States
## 6                        MN              56649 United States
## 7                        RI              02837 United States
## 8                        NJ              07717 United States
## 9                        AK              99517 United States
## 10                       FL              34689 United States
```

```
## 11                       RI              02874 United States
## 12                       MA              02715 United States
## 13                       RI              02818 United States
## 14                       VA              23456 United States
## 15                       MS              39110 United States
## 16                       ID              83605 United States
## 17                       MA              02210 United States
##    Traveling_Library__c Official_Charter_Number__c First_Map_Date__c Map_Me__c
## 1                 FALSE                      76585        2018-08-03      TRUE
## 2                 FALSE                      83568        2020-04-13      TRUE
## 3                 FALSE                      13135        2019-10-01      TRUE
## 4                 FALSE                     107184        2020-09-02      TRUE
## 5                 FALSE                     117921        2021-05-03      TRUE
## 6                 FALSE                       6426        2014-10-10      TRUE
## 7                 FALSE                      26399        2015-09-10      TRUE
## 8                 FALSE                      29517        2015-09-10      TRUE
## 9                 FALSE                      39053        2016-07-05      TRUE
## 10                FALSE                      23484        2016-11-14      TRUE
## 11                FALSE                     135342        2021-10-13      TRUE
## 12                FALSE                     135518        2021-12-13      TRUE
## 13                FALSE                     152156        2022-10-04      TRUE
## 14                FALSE                     131731        2023-02-21      TRUE
## 15                FALSE                     157744        2023-03-01      TRUE
## 16                FALSE                     162828        2023-03-20      TRUE
## 17                FALSE                     160393        2023-04-10      TRUE
##    Map_Date__c Duplicate_Charter_Number__c Count_of_Primary_Stewards__c
## 1   2018-08-03                      FALSE                            1
## 2   2020-04-13                      FALSE                            1
## 3   2019-10-01                      FALSE                            1
## 4   2020-09-02                      FALSE                            1
## 5   2021-05-03                      FALSE                            1
## 6   2019-01-22                      FALSE                            1
## 7   2020-04-01                      FALSE                            1
## 8   2015-09-10                      FALSE                            1
## 9   2016-07-05                      FALSE                            1
## 10  2016-11-14                      FALSE                            1
## 11  2021-10-18                      FALSE                            1
## 12  2021-12-13                      FALSE                            1
## 13  2022-10-04                      FALSE                            1
## 14  2023-02-21                      FALSE                            1
## 15  2023-03-01                      FALSE                            1
## 16  2023-03-20                      FALSE                            1
## 17  2023-04-10                      FALSE                            1
##    Latitude_MapAnything__c Longitude_MapAnything__c
## 1                 29.46048               -81.11792
## 2                 44.21357               -68.71957
## 3                 54.40273                24.03162
## 4                 41.57126               -71.44160
## 5                 41.83050               -71.38526
## 6                 48.58890               -93.46442
## 7                 41.48505               -71.14141
## 8                 40.18972               -74.00997
## 9                 61.20429              -149.92247
## 10                28.14467               -82.78962
```

```
## 11                41.50778                -71.41860
## 12                41.80444                -71.12383
## 13                41.66889                -71.42342
## 14                 0.00000                  0.00000
## 15                 0.00000                  0.00000
## 16                 0.00000                  0.00000
## 17                 0.00000                  0.00000
##    Library_Geolocation__Latitude__s Library_Geolocation__Longitude__s
## 1                          29.46050                          -81.11795
## 2                          44.21276                          -68.71880
## 3                          54.40512                           24.02948
## 4                          41.57137                          -71.44158
## 5                          41.83050                          -71.38526
## 6                          48.58890                          -93.46443
## 7                          41.48536                          -71.14151
## 8                          40.18918                          -74.01003
## 9                          61.20386                         -149.92313
## 10                         28.14444                          -82.78960
## 11                         41.50778                          -71.41860
## 12                         41.80444                          -71.12383
## 13                         41.66889                          -71.42342
## 14                         36.42580                          -75.56300
## 15                         32.46640                           90.15027
## 16                         43.67455                          116.69132
## 17                         42.35282                          -71.04923
##    check_in_count state alt_state
## 1               2  <NA>      <NA>
## 2               0  <NA>      <NA>
## 3               0  <NA>      <NA>
## 4               4  <NA>      <NA>
## 5              17  <NA>      <NA>
## 6               0  <NA>      <NA>
## 7               1  <NA>      <NA>
## 8               0  <NA>      <NA>
## 9               0  <NA>      <NA>
## 10              1  <NA>      <NA>
## 11              4  <NA>      <NA>
## 12              2  <NA>      <NA>
## 13              2  <NA>      <NA>
## 14              0  <NA>      <NA>
## 15              0  <NA>      <NA>
## 16              0  <NA>      <NA>
## 17              0  <NA>      <NA>
```

The library in Alytus, Lithuania is the only one not in the U.S. So let's just drop that one and add the new state assignments:

```
case_1 <- temp_libraries %>% filter(is.na(state) & is.na(alt_state)) %>% filter(!is.na(State_Province_R
  filter(City__c != "Alytus") %>% mutate(state_name = convert_state(State_Province_Region__c), long = L
```

2.

```
case_2 <- temp_libraries %>% filter(!is.na(state) & is.na(alt_state)) %>%
  mutate(state_name = state, long = Library_Geolocation__Longitude__s, lat = Library_Geolocation__Latitu
```

3.

```
case_3 <- temp_libraries %>% filter(is.na(state) & !is.na(alt_state)) %>%
  mutate(state_name = alt_state, long = Longitude_MapAnything__c, lat = Latitude_MapAnything__c)
```

4.

```
case_4 <- temp_libraries %>% filter(state == alt_state) %>% mutate(state_name = state, long = Library_G
```

5.

```
# state != alt_state but the given state code is equal to state
case_5a <- temp_libraries %>% filter(state != alt_state ) %>% mutate(State_Province_Region__c = convert
```

```
# state != alt_state but the given state code is equal to alt_state
case_5b <- temp_libraries %>% filter(state != alt_state ) %>%  mutate(State_Province_Region__c = conver
```

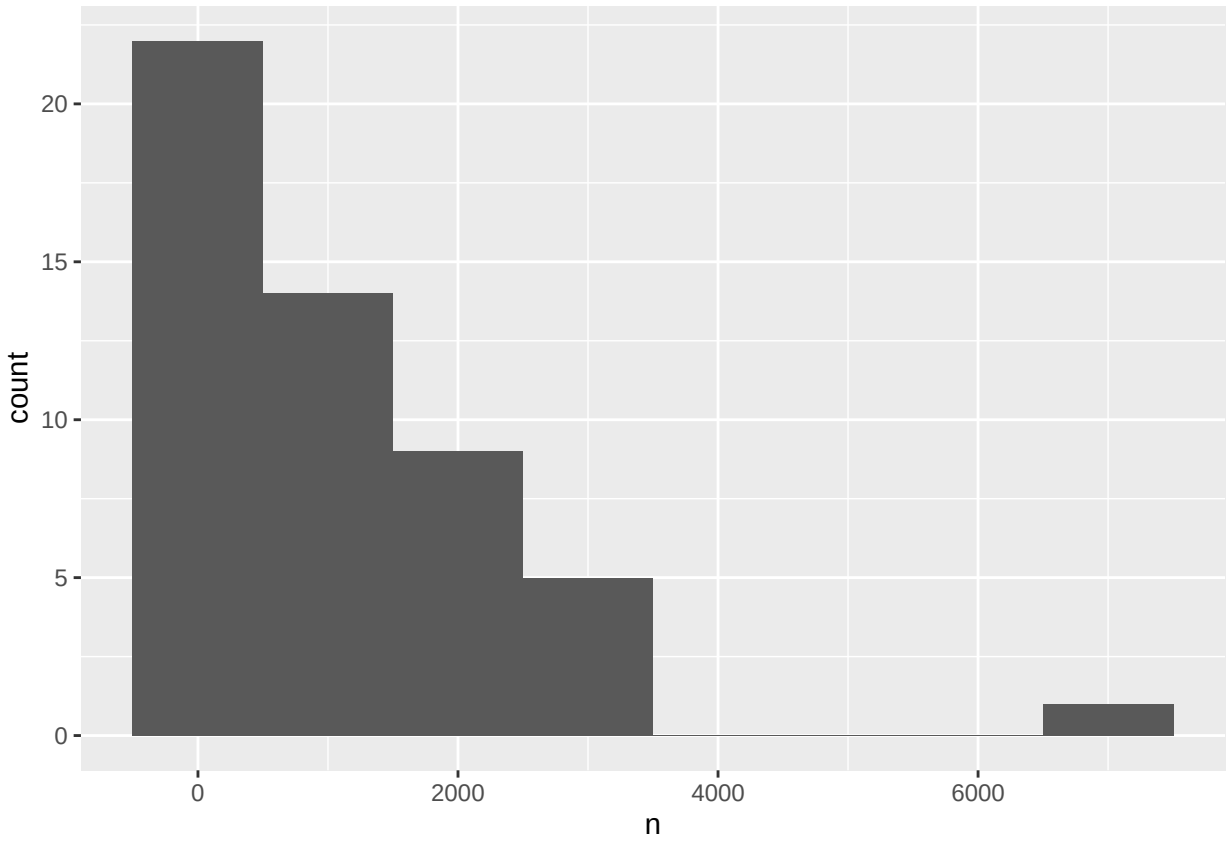
Joining it all together:

```
temp_libraries <- bind_rows(case_1, case_2, case_3, case_4, case_5a, case_5b)
# Drop state, alt_state, and both original coordinates
libraries <- select(temp_libraries, -c(state, alt_state, State_Province_Region__c, Library_Geolocation_
  arrange(id) # Sort rows based off id
```

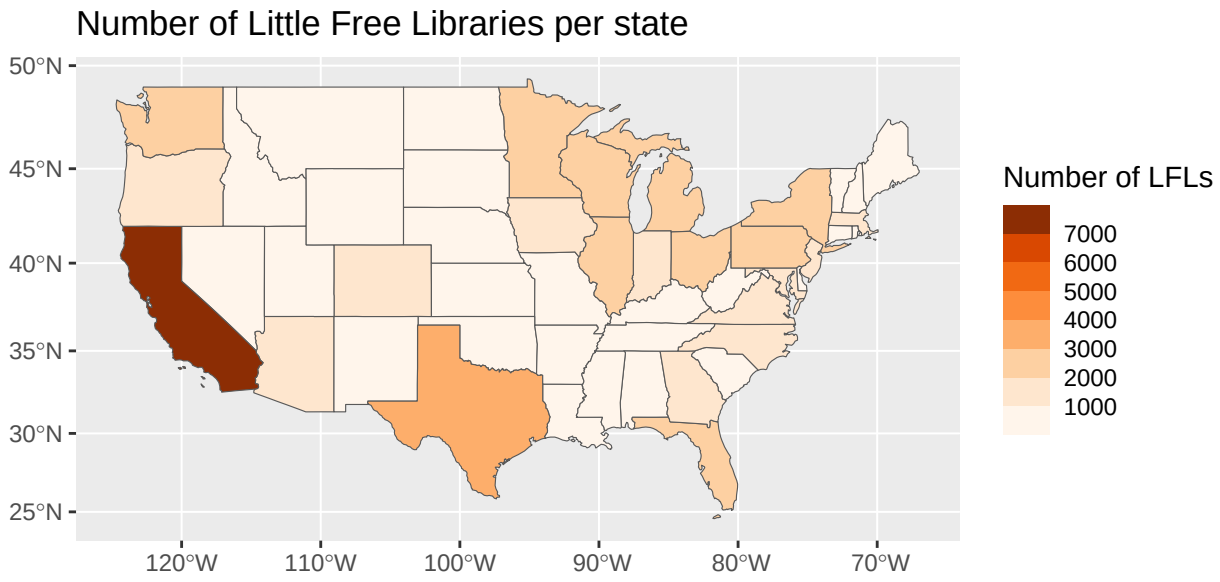
Finally we can do some analysis with states:

```
state_counts <- libraries %>% group_by(state_name) %>% count %>% arrange(desc(n))
state_counts <- state_counts %>% rename("NAME" = "state_name")
```

```
state_counts %>% ggplot(aes(x = n)) +
  geom_histogram(binwidth = 1000)
```



```
st_transform(us_states, crs = 3857) %>% full_join(state_counts, by = join_by(NAME)) %>%
  ggplot(aes(fill = n)) +
  geom_sf() +
  labs(fill = "Number of LFLs") +
  scale_fill_fermenter(n.breaks = 9, direction = 1, palette = "Oranges") +
  ggtitle("Number of Little Free Libraries per state")
```




```
  #ggsave("lflState.png")
```

And then if we normalize by population:

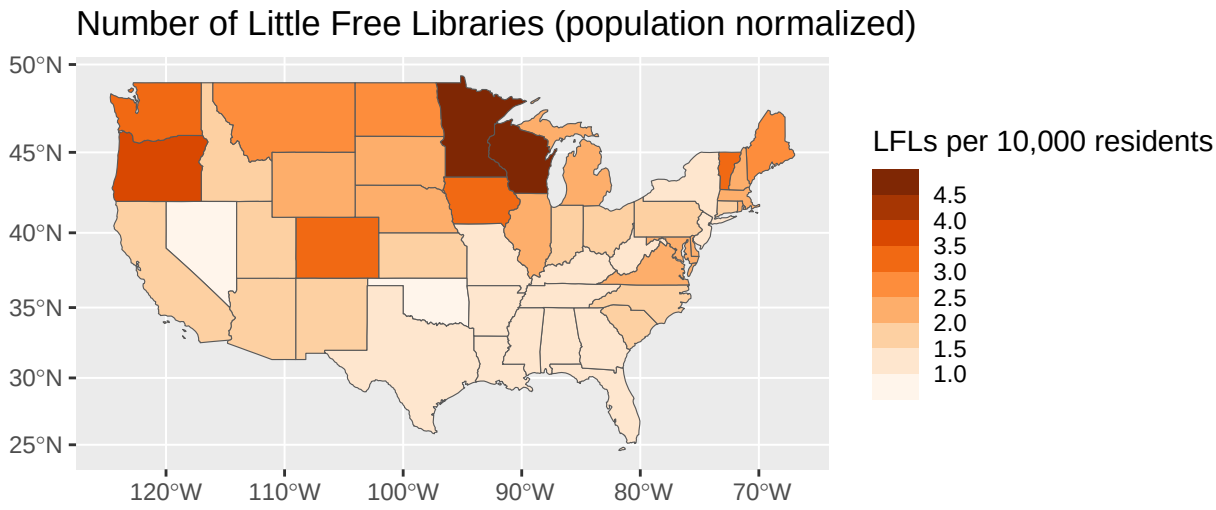
```
state_pops = as_tibble(read_csv("census_pop_data.csv"))
```

```
## Rows: 71 Columns: 53
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Label (Grouping)
## num (52): Alabama, Alaska, Arizona, Arkansas, California, Colorado, Connecti...
##
## 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.
```

```
state_pops <- state_pops[1,] %>% select(-"Label (Grouping)" ) %>% pivot_longer(cols = everything(), name
norm_state_counts <- state_counts %>% left_join(state_pops) %>% mutate(per_cap = 10000*n/pop) %>% selec
```

```
## Joining with `by = join_by(NAME)`
```

```
st_transform(us_states, crs = 3857) %>% left_join(norm_state_counts, by = join_by(NAME)) %>%
  ggplot(aes(fill = per_cap)) +
  geom_sf() +
  labs(fill = "LFLs per 10,000 residents") +
  scale_fill_fermenter(n.breaks = 9, direction = 1, palette = "Oranges") +
  ggtitle("Number of Little Free Libraries (population normalized)")
```




```
#ggsave("lflStateNorm.png")
```

Per capita, it looks like the distribution of LFLs is biased towards the Midwest, in particular Minnesota and Wisconsin. This is not surprising, as the organization was founded in Hudson, Wisconsin and eventually moved to Minneapolis, Minnesota.

Analysis by county

There are many faults to doing geospatial analysis based off zipcodes (they overlap, are not contiguous, etc...). Moreover, in the most recent 2020 census differential privacy techniques applied to data on finer scales resulted in a lot of inaccuracies. So for the purposes of this analysis let's use county(and county-equivalent) data.

```
library(tidycensus)
census_api_key("7ce885ce139c2116d8d26ffca665df473f98a98c")
```

```
## To install your API key for use in future sessions, run this function with `install = TRUE`.
```

```
county_incomes <- get_acs(geography = "county", variables = "B19013_001", year = 2021)
```

```
## Getting data from the 2017-2021 5-year ACS
```

```
county_populations <- get_acs(geography = "county", variables = "B02001_001", year = 2021)
```

```
## Getting data from the 2017-2021 5-year ACS
```

```
county_incomes <- county_incomes %>% full_join(county_populations, by = "NAME")
```

```
# Convert the coordinates to a sf object
# Our coordinate reference system is the WGS84 standard which is what Google
# maps uses. Its EPSG Code is 4326. The format for a point is (longitude, latitude).
lib_pts <- libraries %>% st_as_sf(coords = c("long", "lat"), crs = st_crs(4326))

# Read in and transform the GADM county data to WGS84 format.
GADM_data2 <- st_read(dsn = "gadm36_USA_gpkg/gadm36_USA.gpkg", layer = "gadm36_USA_2")
```

```
## Reading layer `gadm36_USA_2' from data source
##   `/Users/kcrans/Desktop/projects/little_free/gadm36_USA_gpkg/gadm36_USA.gpkg'
##   using driver `GPKG'
## Simple feature collection with 3148 features and 13 fields
## Geometry type: MULTIPOLYGON
```

```
## Dimension:     XY
## Bounding box:  xmin: -179.1506 ymin: 18.90986 xmax: 179.7734 ymax: 72.6875
## Geodetic CRS:  WGS 84
```

```
county_pts <- st_transform(GADM_data2, crs = 4326)

# Differences between GADM and 2021 ACS county definitions means we have to do a lot of manual data cle

county_pts <- county_pts %>% mutate(COUNTY = recode(TYPE_2, "Independent City" = "city", "City And Coun
  mutate(NAME = recode(NAME_2, "Fairfax City" = "Fairfax", "Carson City" = "Carson", "Roanoke City" = "
  mutate(NAME = str_replace_all(NAME, "Saint", "St.")) %>%
  mutate(NAME = str_replace_all(NAME, "De Kalb", "DeKalb")) %>%
  mutate(NAME = str_replace_all(NAME, "La Salle", "LaSalle")) %>%
  unite(county_name, c("NAME", "COUNTY"), sep = " ") %>% mutate(county_name = recode(county_name, "Clif

# Make a data.frame of all the possible county names
county_names <- county_pts$county_name

# Find the intersections between the library points and state polygons
# Convert to an integer to use as an index in the state names data.frame.
classifications <- as.integer(st_intersects(lib_pts, county_pts))

temp_libraries <- libraries %>% mutate(
  county = county_names[classifications])
```

These are all the libraries which did not intersect any of the county boundaries we looked at.

```
temp_libraries %>% filter(is.na(county))
```

```
##       id          Name               Street__c             City__c
## 1  10894 LIB-000034171 1932 S. Oceanshore Blvd       Flagler Beach
## 2  19119 LIB-000046571   319 Dunham Point Road           Deer Isle
## 3  26622 LIB-000053055          1 Newport Ave     North Kingstown
## 4  34172 LIB-000064463      567 Angell Street           Providence
## 5  46526 LIB-000009684    510 Shorewood Drive International Falls
## 6  51495 LIB-000014854      86 Butts Rock Rd.      Little Compton
## 7  51556 LIB-000014915    Ocean Ave & Lincoln     Avon by the Sea
## 8  56156 LIB-000019673     2013 Wildwood Lane            Anchorage
## 9  58958 LIB-000022491         1800 gulf road      Tarpon springs
## 10 63301 LIB-000071612         185 Ferry Road        Saunderstown
## 11 64910 LIB-000073170        261 Hart Street              Dighton
## 12 75691 LIB-000083936     22 Beachwood Drive             Warwick
## 13 78630 LIB-000086854                   <NA>          Sandbridge
## 14 78786 LIB-000087009                   <NA>             Madison
## 15 79239 LIB-000087459   114 W. Chicago Street            Caldwell
## 16 82395 LIB-000090612      64 Sleeper Street              Boston
##    Postal_Zip_Code__c    Country__c Traveling_Library__c
## 1               32136 United States                FALSE
## 2               04627 United States                FALSE
## 3               02852 United States                FALSE
## 4               02906 United States                FALSE
## 5               56649 United States                FALSE
## 6               02837 United States                FALSE
## 7               07717 United States                FALSE
## 8               99517 United States                FALSE
```

```
## 9               34689 United States               FALSE
## 10              02874 United States               FALSE
## 11              02715 United States               FALSE
## 12              02818 United States               FALSE
## 13              23456 United States               FALSE
## 14              39110 United States               FALSE
## 15              83605 United States               FALSE
## 16              02210 United States               FALSE
##    Official_Charter_Number__c First_Map_Date__c Map_Me__c Map_Date__c
## 1                       76585        2018-08-03      TRUE  2018-08-03
## 2                       83568        2020-04-13      TRUE  2020-04-13
## 3                      107184        2020-09-02      TRUE  2020-09-02
## 4                      117921        2021-05-03      TRUE  2021-05-03
## 5                        6426        2014-10-10      TRUE  2019-01-22
## 6                       26399        2015-09-10      TRUE  2020-04-01
## 7                       29517        2015-09-10      TRUE  2015-09-10
## 8                       39053        2016-07-05      TRUE  2016-07-05
## 9                       23484        2016-11-14      TRUE  2016-11-14
## 10                     135342        2021-10-13      TRUE  2021-10-18
## 11                     135518        2021-12-13      TRUE  2021-12-13
## 12                     152156        2022-10-04      TRUE  2022-10-04
## 13                     131731        2023-02-21      TRUE  2023-02-21
## 14                     157744        2023-03-01      TRUE  2023-03-01
## 15                     162828        2023-03-20      TRUE  2023-03-20
## 16                     160393        2023-04-10      TRUE  2023-04-10
##    Duplicate_Charter_Number__c Count_of_Primary_Stewards__c check_in_count
## 1                        FALSE                            1              2
## 2                        FALSE                            1              0
## 3                        FALSE                            1              4
## 4                        FALSE                            1             17
## 5                        FALSE                            1              0
## 6                        FALSE                            1              1
## 7                        FALSE                            1              0
## 8                        FALSE                            1              0
## 9                        FALSE                            1              1
## 10                       FALSE                            1              4
## 11                       FALSE                            1              2
## 12                       FALSE                            1              2
## 13                       FALSE                            1              0
## 14                       FALSE                            1              0
## 15                       FALSE                            1              0
## 16                       FALSE                            1              0
##       state_name       long      lat county
## 1        Florida  -81.11795 29.46050   <NA>
## 2          Maine  -68.71880 44.21276   <NA>
## 3    Connecticut  -71.44158 41.57137   <NA>
## 4   Rhode Island  -71.38526 41.83050   <NA>
## 5      Minnesota  -93.46443 48.58890   <NA>
## 6   Rhode Island  -71.14151 41.48536   <NA>
## 7     New Jersey  -74.01003 40.18918   <NA>
## 8         Alaska -149.92313 61.20386   <NA>
## 9        Florida  -82.78960 28.14444   <NA>
## 10  Rhode Island  -71.41860 41.50778   <NA>
## 11 Massachusetts  -71.12383 41.80444   <NA>
```

```
## 12  Rhode Island  -71.42342 41.66889   <NA>
## 13      Virginia  -75.56300 36.42580   <NA>
## 14   Mississippi   90.15027 32.46640   <NA>
## 15         Idaho  116.69132 43.67455   <NA>
## 16 Massachusetts  -71.04923 42.35282   <NA>
```

These all appear to be locations t0o close to maritime borders for the resolution in our geometry objects.

Manually fix all the county classifications that have changed (i.e. new counties and consolidations)

```
libraries <- temp_libraries  %>% unite(full_name, c("county", "state_name"), sep = ", ")  %>%
  mutate(full_name = recode(full_name, "Shannon County, South Dakota" = "Oglala Lakota County, South Dak
  mutate(full_name = replace(full_name, City__c == "Valdez", "Chugach Census Area, Alaska")) %>%
  mutate(full_name = replace(full_name, City__c == "Kake", "Prince of Wales-Hyder Census Area, Alaska"))
  mutate(full_name = replace(full_name, City__c == "Wrangell", "Wrangell City and Borough, Alaska")) %>%
  mutate(full_name = replace(full_name, City__c == "Sackets Harbor", "Jefferson County, New York")) %>%
  mutate(full_name = replace(full_name, City__c == "Washburn" & Postal_Zip_Code__c == "54891", "Bayfield
  mutate(full_name = replace(full_name, City__c == "Baileys Harbor", "Door County, Wisconsin")) %>%
  mutate(full_name = replace(full_name, City__c == "Duluth" & Postal_Zip_Code__c == "55802", "St. Louis
  mutate(full_name = replace(full_name, City__c == "Sodus Pt.", "Wayne County, New York")) %>%
  mutate(full_name = replace(full_name, City__c == "Bridgman", "Berrien County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Menominee", "Menominee County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Erie" & Postal_Zip_Code__c == "16505", "Erie County
  mutate(full_name = replace(full_name, City__c == "Niagara Falls", "Niagara County, New York")) %>%
  mutate(full_name = replace(full_name, City__c == "Sault Ste. Marie", "Chippewa County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Sturgeon Bay", "Door County, Wisconsin")) %>%
  mutate(full_name = replace(full_name, City__c == "Manistee", "Manistee County, Michigan")) %>%
  mutate(full_name = replace(full_name, Postal_Zip_Code__c == "14612", "Monroe County, New York")) %>%
  mutate(full_name = replace(full_name, City__c == "Houghton", "Houghton County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Mears", "Oceana County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Racine", "Racine County, Wisconsin")) %>%
  mutate(full_name = replace(full_name, City__c == "Lake Linden", "Houghton County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Muskegon", "Muskegon County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "East Tawas", "Iosco County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Fineview", "Jefferson County, New York")) %>%
  mutate(full_name = replace(full_name, City__c == "Augres", "Arenac County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Petoskey", "Emmet County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Omena", "Leelanau County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Sheboygan", "Sheboygan County, Wisconsin")) %>%
  mutate(full_name = replace(full_name, City__c == "Houghton", "Houghton County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "MANITOWOC", "Manitowoc County, Wisconsin")) %>%
  mutate(full_name = replace(full_name, City__c == "Port Huron", "St. Clair County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Detroit", "Wayne County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Wolcott" & Postal_Zip_Code__c == "14590", "Wayne Cou
  mutate(full_name = replace(full_name, City__c == "Tawas City", "Iosco County, Michigan")) %>%
  mutate(full_name = replace(full_name, City__c == "Marinette", "Marinette County, Wisconsin")) %>%

  mutate(full_name = replace(full_name, City__c == "Flagler Beach", "Flagler County, Florida")) %>%
  mutate(full_name = replace(full_name, City__c == "Deer Isle", "Hancock County, Maine")) %>%
  mutate(full_name = replace(full_name, City__c == "North Kingstown", "Washington County, Rhode Island"))
  mutate(full_name = replace(full_name, City__c == "North Kingstown", "Washington County, Rhode Island"))
  mutate(full_name = replace(full_name, City__c == "Providence" & Postal_Zip_Code__c == "02906", "Provid
  mutate(full_name = replace(full_name, City__c == "International Falls", "Koochiching County, Minnesota
  mutate(full_name = replace(full_name, City__c == "Little Compton", "Newport County, Rhode Island")) %>
  mutate(full_name = replace(full_name, City__c == "Avon by the Sea", "Monmouth County, New Jersey")) %>
```

```
  mutate(full_name = replace(full_name, City__c == "Anchorage", "Jefferson County, Kentucky")) %>%
  mutate(full_name = replace(full_name, City__c == "Tarpon springs", "Pinellas County, Florida")) %>%
  mutate(full_name = replace(full_name, City__c == "Saunderstown", "Washington County, Rhode Island")) %
  mutate(full_name = replace(full_name, City__c == "Dighton", "Bristol County, Massachusetts")) %>%
  mutate(full_name = replace(full_name, City__c == "Warwick" & Postal_Zip_Code__c == "02818", "Kent Cou
  mutate(full_name = replace(full_name, City__c == "Sandbridge", "Virginia Beach city, Virginia")) %>%
  mutate(full_name = replace(full_name, City__c == "Madison" & Postal_Zip_Code__c == "39110", "Madison
  mutate(full_name = replace(full_name, City__c == "Caldwell" & Postal_Zip_Code__c == "83605", "Canyon
  mutate(full_name = replace(full_name, City__c == "Boston" & Postal_Zip_Code__c == "02210", "Suffolk C
  mutate(NAME = tolower(full_name))
```

```
county_counts <- libraries %>% group_by(NAME) %>% count %>% arrange(desc(n))
county_incomes <- county_incomes %>% mutate(NAME = tolower(NAME))
county_stats <- county_counts %>% left_join(county_incomes, by = "NAME")
```

```
county_stats <- rename(county_stats, est_income = estimate.x, est_pop = estimate.y, moe = moe.x ) %>% s
```

Top 5 counties by total number of little free libraries:

```
county_stats %>% arrange(desc(n)) %>% head(5)
```

```
## # A tibble: 5 x 5
## # Groups:   NAME [5]
##   NAME                              n est_income  est_pop libs_percap
##   <chr>                         <int>      <dbl>    <dbl>       <dbl>
## 1 los angeles county, california  1337      76367 10019635        1.33
## 2 king county, washington         1062     106326  2240876        4.74
## 3 cook county, illinois            918      72121  5265398        1.74
## 4 hennepin county, minnesota       912      85438  1270283        7.18
## 5 san diego county, california     799      88240  3296317        2.42
```

Top 5 counties by number of little free libraries adjusted for population:

```
county_stats %>% arrange(desc(libs_percap)) %>% head(5)
```

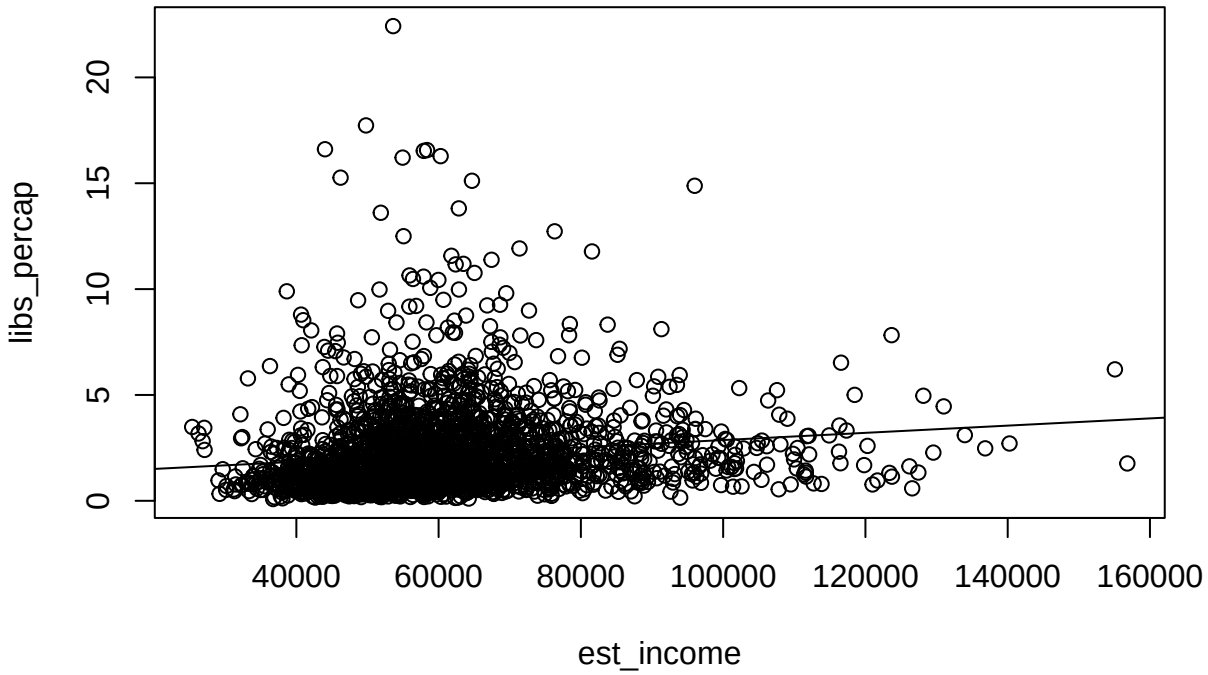
```
## # A tibble: 5 x 5
## # Groups:   NAME [5]
##   NAME                            n est_income est_pop libs_percap
##   <chr>                       <int>      <dbl>   <dbl>       <dbl>
## 1 sherman county, oregon          4      53606    1784        22.4
## 2 daggett county, utah            1      49792     564        17.7
## 3 lewis county, idaho             6      44028    3613        16.6
## 4 jerauld county, south dakota    3      58375    1811        16.6
## 5 banner county, nebraska         1      57917     605        16.5
```

```
summary(lm(libs_percap ~ est_income, data = county_stats))
```

```
##
## Call:
## lm(formula = libs_percap ~ est_income, data = county_stats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7334 -1.1971 -0.5822  0.5199 20.3427
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 1.165e+00  1.588e-01   7.337 2.97e-13 ***
## est_income  1.705e-05  2.544e-06   6.701 2.57e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.994 on 2438 degrees of freedom
## Multiple R-squared:  0.01808,    Adjusted R-squared:  0.01768
## F-statistic:  44.9 on 1 and 2438 DF,  p-value: 2.569e-11
```

```
plot(libs_percap ~ est_income, data = county_stats)
abline(lm(libs_percap ~ est_income, data = county_stats))
```



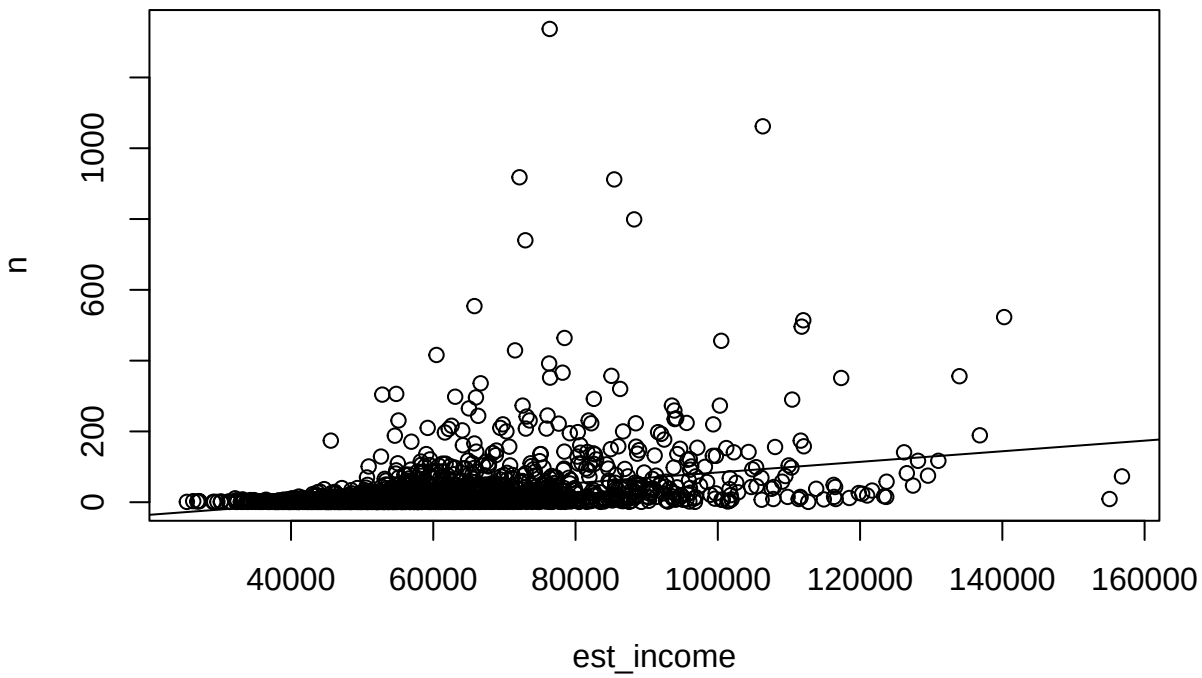
So there is a low correlation between average income and number of little free libraries per capita. If we ignore the adjustment for population, we get:

```
summary(lm(n ~ est_income, data = county_stats))
```

```
##
## Call:
## lm(formula = n ~ est_income, data = county_stats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -157.53  -20.55   -8.02    4.37 1288.32
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.567e+01  5.208e+00  -12.61   <2e-16 ***
## est_income   1.497e-03  8.342e-05   17.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65.37 on 2438 degrees of freedom
## Multiple R-squared:  0.1167, Adjusted R-squared:  0.1164
```

```
## F-statistic: 322.2 on 1 and 2438 DF,  p-value: < 2.2e-16
```

```
plot(n ~ est_income, data = county_stats)
abline(lm(n ~ est_income, data = county_stats))
```



Note that incomes can very significantly within a county(there can be rich and poor cites, rich and poor neighborhoods, etc...) The analysis would be significantly improved if we used income figures on a finer-grained scale.