Telematics Data 101

Parunjodhi Munisamy, Hanna Lee, Tulip Daaboul, Amber Liu, Yaretsy Castro

December 14, 2023

Overview of Telematics data

Telematics data (https://www.verizonconnect.com/resources/article/what-is-telematics/) essentially comes from the telematics system found in a vehicle's tracking equipment. It records data that is being sent, received or stored by the tracking system. More specifically, telematics data include data on speed, precise location (addresses or coordinates) and time, route undertaken, safety metrics, acceleration, fuel intake, and many more. Telematics data can also include multiple datasets that each consists of a subset of variables. Such an example will be explained below using the data used to create this report.

But why should one care about telematics data? Well, telematics data is important for a myriad of reasons. For one, vehicles, especially cars, are not going anywhere anytime soon. Therefore, any data collected on their performance can help identify potential ways in which not only vehicles can operate more efficiently but also how the transport system can be more efficient. This can help reduce traffic congestion as well as fuel consumption. With clean energy and the energy transition becoming a priority for many key stakeholders from policymakers to drivers, it is more important now than ever to create more sustainable transportation systems. Telematics data could definitely help in pushing this endeavor forward.

Another crucial element of using telematics data is to find patterns in terms of safety on roads. Pinning down potential unsafe routes or even unsafe times can inform authorities and policymakers to ensure the safety of everyone. From a technological point of view, the more telematics data gathered, the more effective existing tracking systems as well as GPS systems can be. With AI technology being on the rise, it is only a matter of time that AI is implemented in vehicle systems, including tracking devices. Telematics data could be a potential source of data that could inform structure and build these new AI models to make traveling safer and faster.

The automotive industry uses telematics data. Insurance companies use it as a means to monitor a driver's behavior. It can record harsh events and measure safety scores. Ride share apps use it for vehicle tracking. Car companies can also use it for maintenance improvements. It would measure when maintenance is needed based on engine hours, miles driven and more. Currently it is being used for fleet management, as in managing a number of vehicles to run on time, within budget and at maximum efficiency.

All about the data

We were provided with eight microtransit tables that contain data on organization associated with the ride, trip requests made by riders, general route information together with harsh events and safety details, and vehicle information with diagnostics and locations. For the purpose of our analysis, we mainly focused on trip requests and vehicle location tables.

Both the trip request table and the vehicle location table contains data spanning from December 2020 to February 2023. The trip request table contains general information about a requested trip including its scheduled and requested time, pick up and drop off time and locations, the completion status, estimated miles and fare. The vehicle location table contains real-time location information for full trips– tracking of where the vehicle was including latitude and longitude as well as its speed during the trip.

Loading the data

```
#R version 4.3.1 (2023-06-16)
#Platform: x86_64-pc-linux-gnu (64-bit)
#Running under: Ubuntu 20.04.1 LTS
#Used remote Smith RStudio

library(tidyverse) #tidyverse_2.0.0
library(tidycensus) #tidycensus_1.5
library(tidygeocoder) #tidygeocoder_1.0.5
library(sf) #sf_1.0-14
library(stringr) #stringr_1.5.0
library(plotly) #plotly_4.10.2
library(sfheaders) #sfheaders_0.4.3
library(leaflet) #leaflet_2.1.2
library(leaflet.extras2) #leaflet.extras2_1.2.2
library(htmlwidgets) #htmlwidgets_1.6.2
```

Loading Telematics data

```
Trip_request <- read.csv("data/TRIP_REQUEST_202308141518.csv")
vehicle_location_data <- read_csv("data/VEHICLE_LOCATION_202308141525.csv")
```

```
## coordinates for ALL pick up and drop off to make coordinates
trip_data_full <- Trip_request %>%
  geocode(DROPOFF_ADDRESS, method = 'census', lat = lat_drop, long= long_drop) %>%
  geocode(PICKUP_ADDRESS, method = 'census', lat = lat_pickup, long= long_pickup) %>%
  drop_na()
```

Loading Census data

```
# variables for 2020 and 2022 census data
var_acs20 <- load_variables(2020, "acs5", cache = TRUE)

var_acs21 <- load_variables(2021, "acs1", cache = TRUE)

var_acs22 <- load_variables(2022, "acs1", cache = TRUE)
```

```
#getting median income data from acs for 2021
median_income <- get_acs(geography = "county subdivision",
              variables = c(medincome = "B19013_001"),
              state = "OH",
              year = 2021,
              geometry = TRUE) %>%
  drop_na()
```

```
#getting public transport data from census for 2021
public_transport <- get_acs(geography = "county subdivision",
              variables = c(public_route = "B08006_008"),
              state = "OH",
              year = 2021,
              geometry = TRUE) %>%
  drop_na()
```

```
# finding median income levels in 2020 in ohio
income_ohio2020 <-
    get_acs(geography = "county",
            state = "OH",
            variables = c(medincome = "B19013_001"),  # median household income
                        #  "B01003_001"), # total pop
            year = 2020,
            geometry = TRUE)%>%
        rename(median_household_income20 = estimate)%>%
        rename(margin_of_error = moe) %>%
       mutate(margin_of_error= coalesce(margin_of_error, 0)) %>%
  select(GEOID, NAME,median_household_income20,geometry)
```

```
# finding poverty levels in 2020 in ohio
povertyrates_ohio2020 <-
      get_acs(geography = "state",
          #persons whose income in last12 months was below poverty level
           variables = "B17001_002",
            # person whose poverty level is tracked
          summary_var = 'B17001_001',
          #  state = "OH",
            geometry = TRUE,
            year = 2020) %>%
        rename(population = summary_est) %>%
  filter(population>0)%>%
  mutate(pov_rate = estimate/population) %>%
  mutate(pov_rate = pov_rate*100) %>%
   select(NAME, population, pov_rate)
```

Navigating obstacles

Loading Telematics data

While loading our data into R we faced multiple difficulties. Our dataset included a total of 8 tables. Some were small, while others were too large. The largest table, TRIP_REQUEST_202308141518, is around 150 MB. This data set took around 30 minutes to load into our local computers. We attempted to upload our data into our GitHub repository, but it turns out we cannot upload any files larger than 25 MB. To solve this issue, we extracted a sample to analyze, however it was not a random sample. We then attempted to put our file into. gitignore, but still had the issue that it was too time consuming. We talked to our sponsors about this issue. 99pLabs gave us two potential solutions to our resource issue; 1. Extract a random sample, 2. Use a remote computer. We were leaning into a random sample, since we thought we did not have access to a remote computer. Turns out, Smith College provides an online RStudio. Once we moved our directory to online, we were able to work with this data set more efficiently.

Geocoding Telematics data

In order to map pickup and drop off locations, we had to geocode the addresses. In our code, we first used method = osm. Using this, it took us around two hours to geocode all of the addresses in the dataset. In addition, our laptops had to be open and connected to wifi. If our laptop closes or we lose connection, the code fails and we would have to restart it. Once we changed our method to census and we started using the rstudio2 server, our code ran in less than 10 minutes and we had our addresses geocoded.

Additionally, using the ‘census’ method allowed us to geocode the locations without acquiring a specific API key, which other methods required. We saved all the latitude and longitude addresses into a datafile, which could then be visualized using the leaflet interactive mapping package. For our visualizations and data analysis, we focused the location scope to addresses in Ohio. Because Honda’s headquarters are in Ohio, most of the addresses were located there.

Applications of Telematics data

Visualizing telematics data can be a challenging task. Besides encountering the above problems in loading and cleaning the data, understanding the tables to be able to figure out what variables we want to plot has proven to be difficult as well. It is easy to plot variables to get a map but creating a meaningful map that can be part of a larger storyline is easier said than done.

Given that the data dictionary for the dataset provided were not very comprehensive, our team had to make assumptions when interpreting the different variables and the unit of observations. Even when filtering the data for only Ohio datapoints, some coordinates outside of these parameters managed to slip through. We suspect it has something to do with how the data was collected and stored rather than the code used to filter the tables

Nevertheless, we were able to come up with multiple data visualizations, both static and interactive, to display specific data points in a purposeful manner.

Data Visualization 1

Trips in Ohio for a particular day

```
#creating dataset for one day from vehicle location dataset
vehicle_location_data_one_day <- vehicle_location_data %>%
  filter(str_detect(EVENT_TIMESTAMP, "2021-08-03"))

#splitting EVENT_TIMESTAMP into day and time for further analysis
vehicle_location_data_one_day[c("date", "time")] <- str_split_fixed(vehicle_location_data_one_da
y$EVENT_TIMESTAMP, ' ', 2)

vehicle_location_data_one_day$month <- format(as.Date(vehicle_location_data_one_day$date, format
="%Y-%m-%d"),"%m")

vehicle_location_data_one_day$year <- format(as.Date(vehicle_location_data_one_day$date, format
="%Y-%m-%d"),"%Y")

vehicle_location_data_one_day$day <- format(as.Date(vehicle_location_data_one_day$date, format
="%Y-%m-%d"),"%d")
```

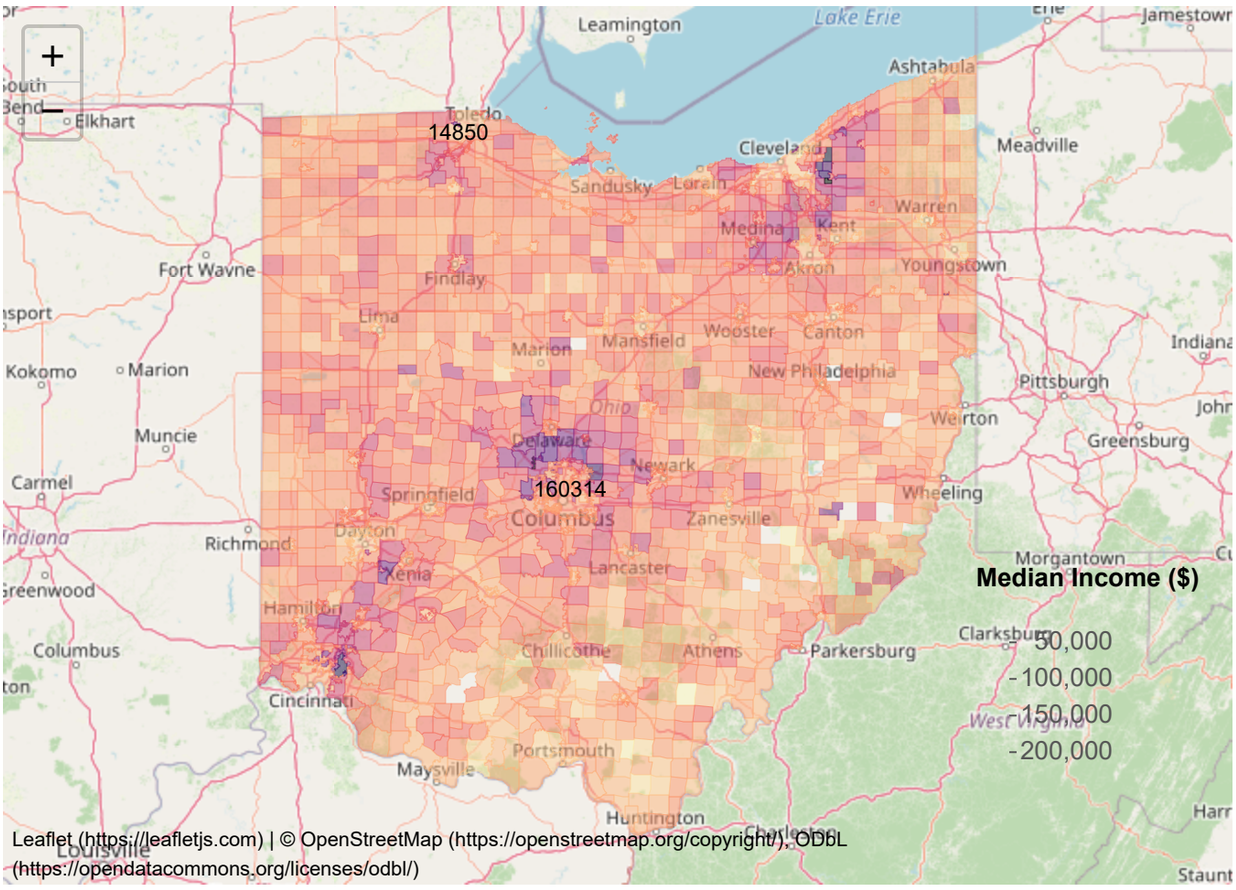
```
# creating color palette to map census data
pal <- colorNumeric(
  palette = "magma",
  domain = median_income$estimate,
  reverse = TRUE
)

pal(c(10, 20, 30, 40, 50))
```

```
## [1] "#808080" "#808080" "#808080" "#808080" "#808080"
```

```
#clustering all trips for one day
map_1 <- leaflet() %>%
  #adding OpenStreetMap
  addTiles() %>%
  #adding census data on income as backdrop
  addPolygons(data = median_income,
              color = ~pal(estimate),
              weight = 0.5,
              smoothFactor = 0.2,
              fillOpacity = 0.5,
              label = ~estimate) %>%
  addLegend(
    position = "bottomright",
    pal = pal,
    values = median_income$estimate,
    title = "Median Income ($)"
  ) %>%
  #adding datapoints for all trips on one day
  addMarkers(
    data = vehicle_location_data_one_day,
    lng = ~LNG,
    lat = ~LAT,
    label = ~VEHICLE_ID,
    popup = ~VEHICLE_ID,
    clusterOptions = markerClusterOptions()
  )

map_1
```



By zooming in and out, the viewer can see the areas drivers took their passengers through on that particular day. The trips are mainly concentrated in Columbus and northern Ohio. This plot is meant to show at a first glance the concentration of trips in Ohio.

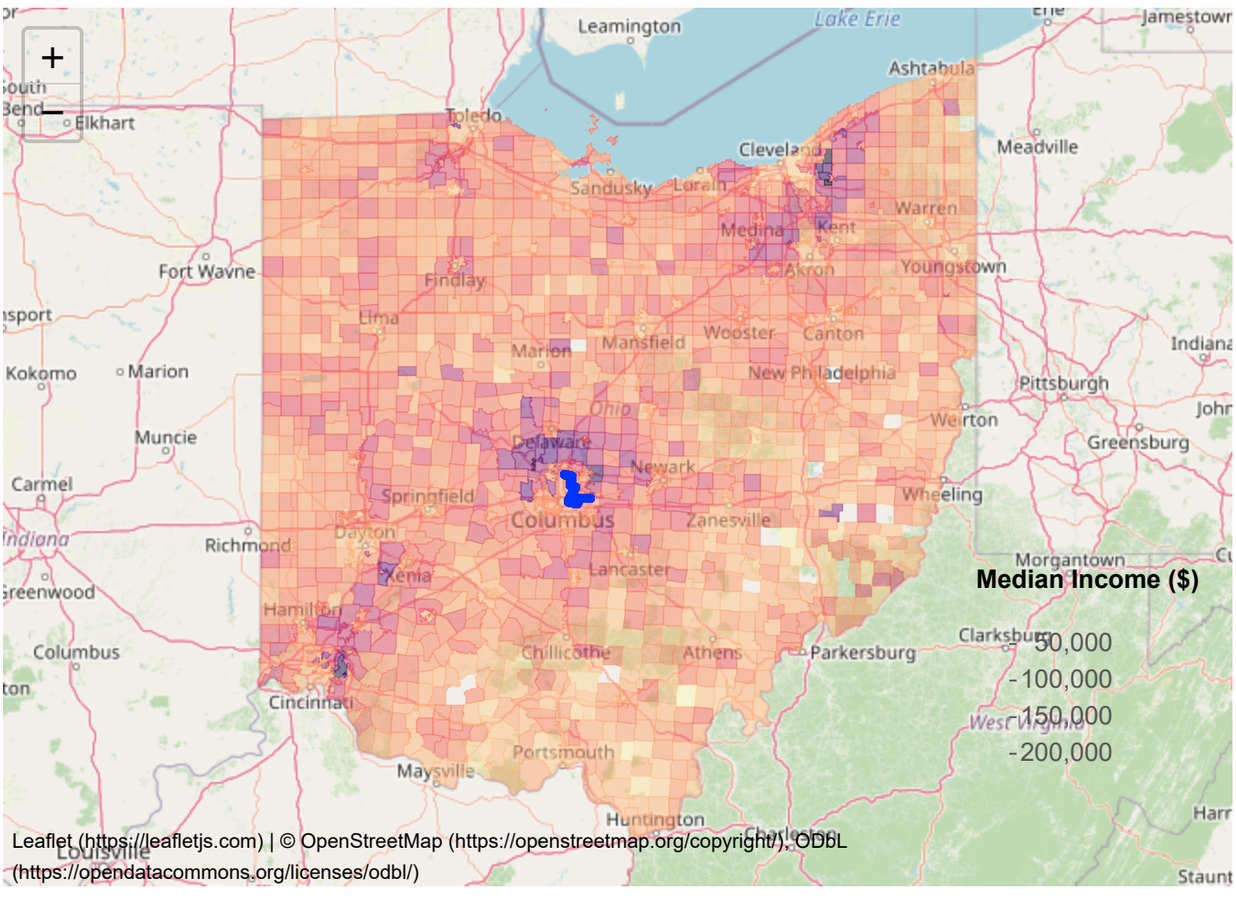
Data Visualization 2

Routes taken on a particular day for one vehicle id

```
#creating dataset for one vehicle id for one day
vehicle_location_data_id_71 <- vehicle_location_data_one_day %>%
  filter(VEHICLE_ID == 71) %>%
  filter(str_detect(EVENT_TIMESTAMP, "2021-08-03"))

#mapping route for one vehicle id for one day
map_2 <- leaflet() %>%
  #adding OpenStreetMap
  addTiles() %>%
  #adding census data on income as backdrop
  addPolygons(data = median_income,
              color = ~pal(estimate),
              weight = 0.5,
              smoothFactor = 0.2,
              fillOpacity = 0.5,
              label = ~estimate) %>%
  addLegend(
    position = "bottomright",
    pal = pal,
    values = median_income$estimate,
    title = "Median Income ($)"
  ) %>%
  #limit to first 3000 rows to decrease processing time
  addPolylines(data = head(vehicle_location_data_id_71, 3000), lng = ~LNG, lat = ~LAT, group = ~
VEHICLE_ID)
  #leaflet.extras2::addPlayback(data = head(vehicle_location_data_id_71, 3000),
                               #time = "time",
                               #options = leaflet.extras2::playbackOptions(speed = 0.5))

map_2
```

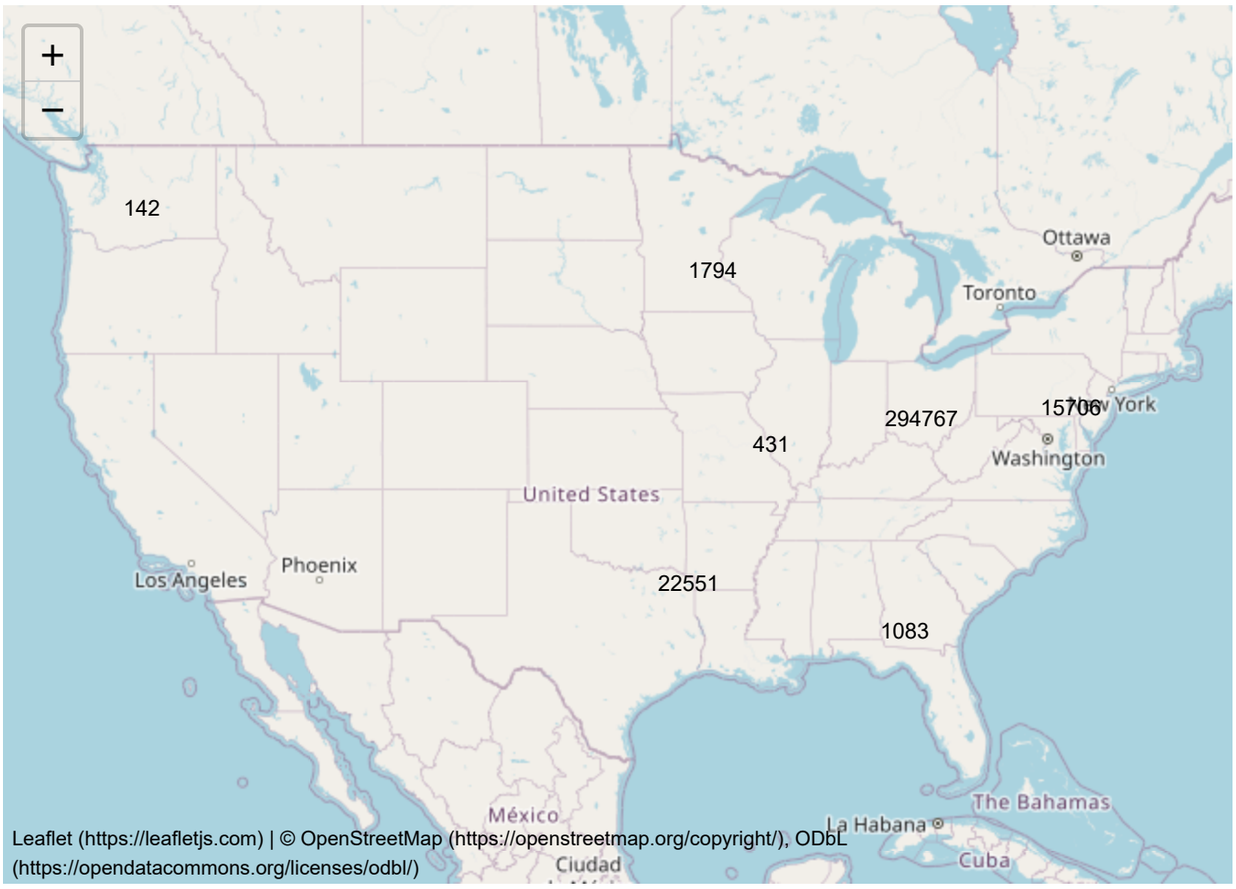


Data Visualization 3

Clustering Pick Up Requests in Trip Request Data

```
triprequest_map <- leaflet(trip_data_full) %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lat = ~lat_pickup, lng = ~long_pickup, clusterOptions = markerClusterOptions()
)

triprequest_map
```



Data Visualization 4

Median income in ohio in 2020

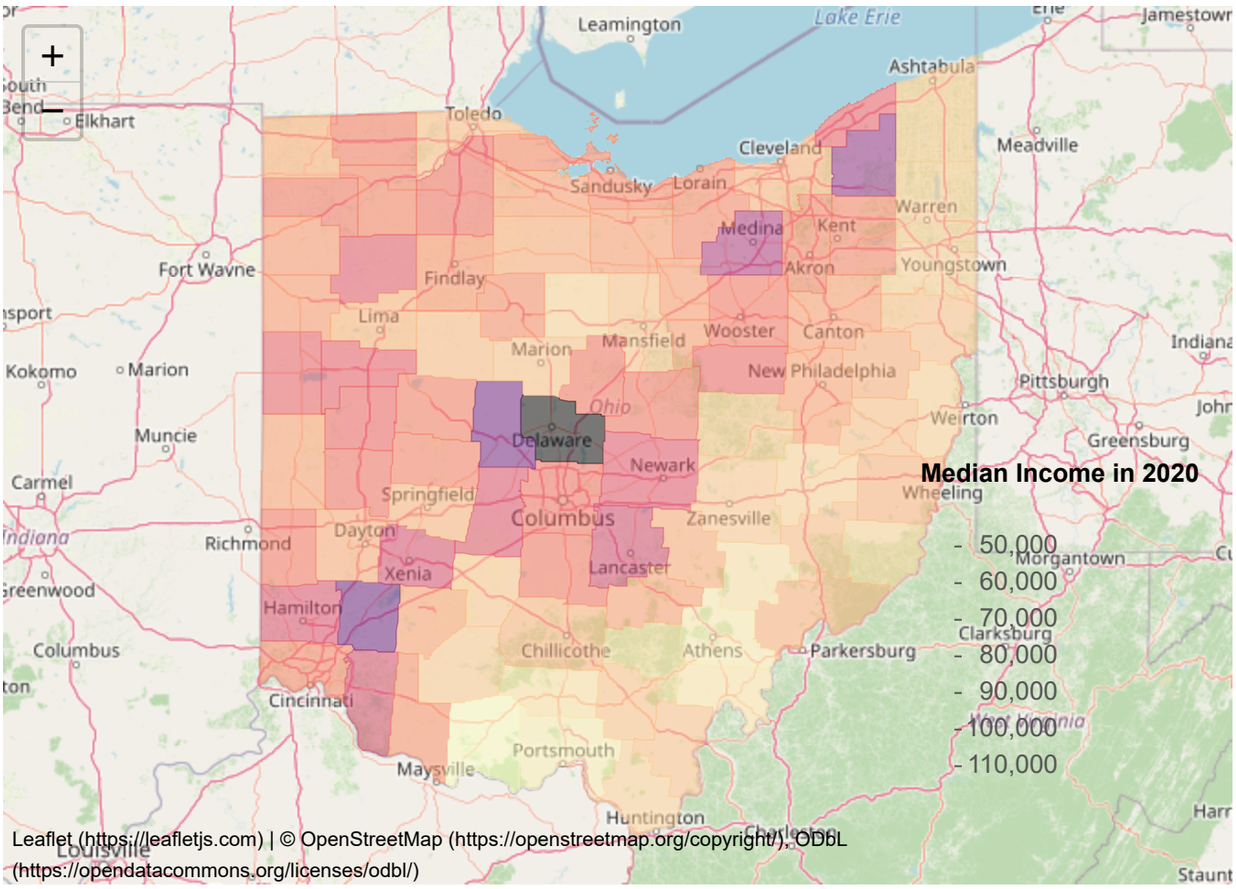
```
# maping the data and making a pallette
pal <- colorNumeric(
  palette = "magma",
  domain = income_ohio2020$median_household_income20,
  reverse = TRUE
)

pal(c(10, 20, 30, 40, 50))
```

```
## [1] "#808080" "#808080" "#808080" "#808080" "#808080"
```

```
med_map_income_20<-leaflet() %>%
  addTiles() %>%
  addPolygons(data = income_ohio2020,
              color = ~pal(median_household_income20),
              weight = 0.5,
              smoothFactor = 0.2,
              fillOpacity = 0.5,
              label = ~median_household_income20) %>%
  addLegend(
    position = "bottomright",
    pal = pal,
    values = income_ohio2020$median_household_income20,
    title = "Median Income in 2020"
  )

med_map_income_20
```



Data Visualization 5

Poverty levels in ohio in 2020

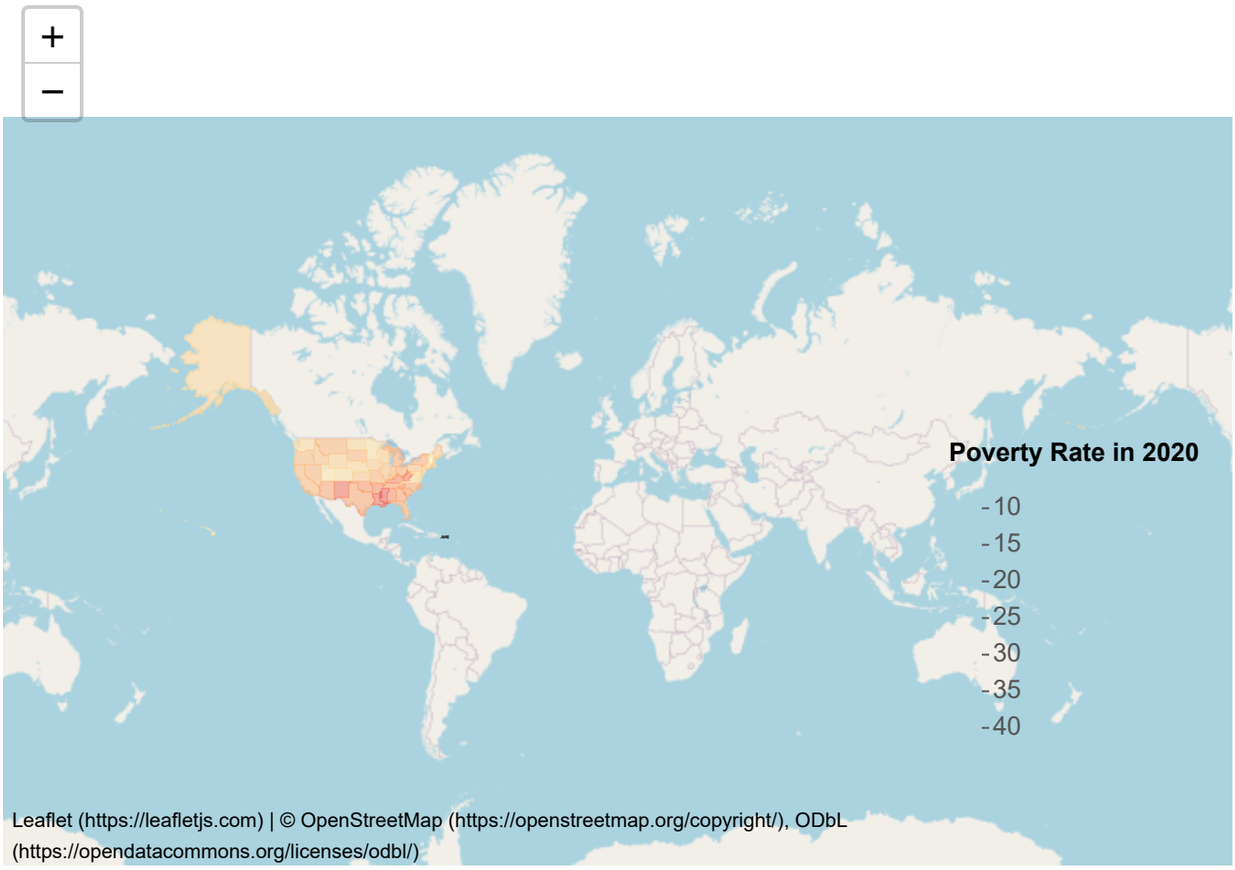
```
# maping the poverty rates in 2020 data
pal2 <- colorNumeric(
  palette = "magma",
  domain = povertyrates_ohio2020$pov_rate,
  reverse = TRUE
)

pal2(c(10, 20, 30, 40, 50))
```

```
## [1] "#FDDB9D" "#ED5A5F" "#812581" "#130D34" "#808080"
```

```
pov_map_rate20<-leaflet() %>%
  addTiles() %>%
  addPolygons(data = povertyrates_ohio2020,
              color = ~pal2(pov_rate),
              weight = 0.5,
              smoothFactor = 0.2,
              fillOpacity = 0.5,
              label = ~pov_rate) %>%
  addLegend(
    position = "bottomright",
    pal = pal2,
    values = povertyrates_ohio2020$pov_rate,
    title = "Poverty Rate in 2020"
  )

pov_map_rate20
```



Appendix

Session Info

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.1 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] htmlwidgets_1.6.2     leaflet.extras2_1.2.2 leaflet_2.2.0
##  [4] sfheaders_0.4.3       plotly_4.10.2         sf_1.0-14
##  [7] tidygeocoder_1.0.5    tidycensus_1.5        lubridate_1.9.2
## [10] forcats_1.0.0         stringr_1.5.0         dplyr_1.1.3
## [13] purrr_1.0.2           readr_2.1.4           tidyr_1.3.0
## [16] tibble_3.2.1          ggplot2_3.4.3         tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.4       xfun_0.40          bslib_0.5.1        tigris_2.0.4
##  [5] tzdb_0.4.0         crosstalk_1.2.0    vctrs_0.6.3        tools_4.3.1
##  [9] generics_0.1.3     curl_5.1.0         parallel_4.3.1     proxy_0.4-27
## [13] fansi_1.0.4        pkgconfig_2.0.3    KernSmooth_2.23-22 data.table_1.14.8
## [17] RColorBrewer_1.1-3 uuid_1.1-0         lifecycle_1.0.3    farver_2.1.1
## [21] compiler_4.3.1     munsell_0.5.0      htmltools_0.5.6    class_7.3-22
## [25] sass_0.4.7         yaml_2.3.7         lazyeval_0.2.2     pillar_1.9.0
## [29] crayon_1.5.2       jquerylib_0.1.4    ellipsis_0.3.2     classInt_0.4-9
## [33] cachem_1.0.8       viridis_0.6.4      mime_0.12          tidyselect_1.2.0
## [37] rvest_1.0.3        digest_0.6.33      stringi_1.7.12     fastmap_1.1.1
## [41] grid_4.3.1         colorspace_2.1-0   cli_3.6.1          magrittr_2.0.3
## [45] utf8_1.2.3         e1071_1.7-13       withr_2.5.1        scales_1.2.1
## [49] rappdirs_0.3.3     bit64_4.0.5        timechange_0.2.0   rmarkdown_2.24
## [53] httr_1.4.7         bit_4.0.5          gridExtra_2.3      hms_1.1.3
## [57] evaluate_0.21      knitr_1.43         viridisLite_0.4.2  rlang_1.1.1
## [61] Rcpp_1.0.11        glue_1.6.2         DBI_1.1.3          xml2_1.3.5
## [65] vroom_1.6.3        rstudioapi_0.15.0  jsonlite_1.8.7     R6_2.5.1
## [69] units_0.8-3
```