[Luke Whyte](https://github.com/lukewhyte" \t "_blank) posted an [article](https://towardsdatascience.com/fast-static-d3-maps-built-with-turf-js-and-the-command-line-5b7c72b7e775) (apologies for a Medium link) over on Towards Data Science showing how to use a command line workflow involving curl, node and various D3 libraries and javascript source files to build a series of SVG static maps. It’s well written and you should give it a read especially since he provides the code and data.

We can do all of that in R with the help of a couple packages and by using a free geocoding service which will also allow us to put more data on the map, albeit with some extra work due to it returning weird values for Hawaii locations.

library(albersusa) # [git.sr.ht/~hrbrmstr/albersusa](http://git.sr.ht/~hrbrmstr/albersusa) | git[la|hu]b/hrbrmstr/albersusa

library(rgeocodio) # [git.sr.ht/~hrbrmstr/rgeocodio](http://git.sr.ht/~hrbrmstr/rgeocodio) | git[la|hu]b/hrbrmstr/rgeocodio

library(tidyverse)

# the data url from the original blog

fil <- "<https://query.data.world/s/awyahzfiikyoqi5ygpvauqslwmqltr>"

read\_csv(fil, col\_types = "cd") %>%

select(area=1, pct=2) %>%

mutate(pct = pct/100) -> xdf # make percents proper percents

gc <- gio\_batch\_geocode(xdf$area)

The result of the geocoding is a data frame that has various confidences associated with the result. We’ll pick the top one for each and then correct for the errant Hawaii longitude it gives back:

map2\_df(gc$query, gc$response\_results, ~{

out <- .y[1,,]

out$area <- .x

out

}) %>%

filter(![is.na](http://is.na)(location.lat)) %>%

select(area, state = address\_components.state, lat=location.lat, lon=location.lng) %>%

mutate(

lat = ifelse(grepl("Honolu", area), 21.3069, lat),

lon = ifelse(grepl("Honolu", area), -157.8583, lon)

) %>%

left\_join(xdf) %>%

as\_tibble() -> area\_pct

area\_pct

## # A tibble: 47 x 5

## area state lat lon pct

##

## 1 McAllen-Edinburg-Mission, TX TX 26.2 -98.1 0.102

## 2 Houston-The Woodlands-Sugar Land, TX TX 29.6 -95.8 0.087

## 3 Santa Maria-Santa Barbara, CA CA 34.4 -120. 0.081

## 4 Las Vegas-Henderson-Paradise, NV NV 36.1 -115. 0.08

## 5 Los Angeles-Long Beach-Anaheim, CA CA 33.9 -118. 0.075

## 6 Miami-Fort Lauderdale-West Palm Beach, FL FL 26.6 -80.1 0.073

## 7 Dallas-Fort Worth-Arlington, TX TX 33.3 -98.4 0.069

## 8 Washington-Arlington-Alexandria, DC-VA-MD-… WV 39.2 -81.7 0.068

## 9 Bridgeport-Stamford-Norwalk, CT CT 41.3 -73.1 0.067

## 10 San Jose-Sunnyvale-Santa Clara, CA CA 37.4 -122. 0.065

## # … with 37 more rows

The albersusa package provides base maps with Alaska & Hawaii elided into a composite U.S. map. As such, we need to elide any points that are in Alaska & Hawaii:

us <- usa\_composite()

us\_map <- fortify(us, region="name")

hi <- select(filter(area\_pct, state == "HI"), lon, lat)

(hi <- points\_elided(hi))

area\_pct[area\_pct$state == "HI", c("lon", "lat")] <- hi

Then, it’s just a matter of using ggplot2:

ggplot() +

geom\_map(

data = us@data, map=us\_map,

aes(map\_id=name),

fill = "white", color = "#2b2b2b", size = 0.1

) +

geom\_point(

data = area\_pct, aes(lon, lat, size = pct),

fill = alpha("#b30000", 1/2), color = "#b30000", shape=21

) +

ggalt::coord\_proj(us\_laea\_proj) +

scale\_y\_continuous(expand=c(0, 3)) +

scale\_radius(

name = NULL, label = scales::percent\_format(1)

) +

labs(x = "Estimated percent of undocumented residents in U.S. metro areas. Source: Pew Research Center") +

theme\_minimal() +

theme(axis.text = element\_blank()) +

theme(axis.title.x = element\_text(hjust=0.5, size = 8)) +

theme(axis.title.y = element\_blank()) +

theme(panel.grid = element\_blank()) +

theme(legend.position = c(0.9, 0.3)) -> gg

ggsave(filename = "map.svg", device = "svg", plot = gg, height = 5, width = 7)

Unlike the post’s featured image (which has to be a bitmap…grrr) the resultant SVG is below:

**FIN**

There is absolutely nothing wrong with working where you’re most comfortable and capable and Luke definitely wields the command line and javascript incredibly well. This alternate way of doing things in R may help other data journalists who are more comfortable in R or want to increase their R knowledge replicate and expand upon Luke’s process.

If you’ve got alternate ways of doing this in R or even (gasp) Python, drop a note in the comments with a link to your blog so folks who are comfortable neither in R nor the command line can see even more ways of producing this type of content.