Shoe Carnival Store Locations

Jeffrey Sumner

March 14, 2019

Introduction

This RMarkdown document will take you through a few examples of R capabilities as well as showcase some of my programming skills. We will break this into three different sections: (1) Web Scraping, (2) Geocoding and (3) Location Analysis. Each section will have a brief demo of a chunk of R code and an explanation behind each piece.

Let's get started!

Methodology

We will be scraping the Shoe Carnival website:
https://stores.shoecarnival.com/
We use **rvest** to dig into the webpage and pull out key tags. From there we use **ggmap** to geocode the scraped locations. **ggmap** will also be used to create all map visuals. **ggmap** (@Article) requires a Google Cloud API key which can be obtained for free BUT billing must be enabled. There is a $200 credit each month which equates to roughly 40,000 locations geocoded.

(1) Web Scraping

To scrape the data we need:
1) Our base URL: **https://stores.shoecarnival.com/**
2) Find all States via the URL above and abbreviate the full state name
3) With the abbreviated state name we can access the following example URL: **https://stores.shoecarnival.com/al/**
4) Once we access this URL we need ALL cities associated with Shoe Carnival; the format of the city url is as follows:
https://stores.shoecarnival.com/ar/northlittlerock/
5) Once at the webpage for each city via the calculated URL, we need to extract the address. The css tag can be obtained using the Google Chrome extension **selector gadget** or by inspecting the source code.

After working through the steps above and performing some minor cleaning techniques to the data, I was able to create a nice dataset of Shoe Carnival addresses.

Let's see how it all works and put it into action ourselves!

Required R Packages

```r
# likely the most useful package in all of R
# suite of packages for data manipulation, visualization, etc.
# install.packages("tidyverse") # run if tidyverse is not installed already
library(tidyverse)
# rvest is the package that we will use to scrape the web data
library(rvest)
# openintro will be used to conver state names to abbreviations for the URL
# install.packages("openintro") # run if openintro is not installed already
library(openintro)
```

Great, we have our packages loaded in via library() and now we are ready to get to the good parts.

Scraping the Data

As mentioned above in the steps, we first need to figure out each state Shoe Carnival is located in to adjust the URL code. Again, we will be use: **https://stores.shoecarnival.com/** to get this information. At the bottom of the URL there is a full list of states. We will be extracting each.

Getting the States

```
sc.url <- read_html("https://stores.shoecarnival.com/")
sc.node <- ".map-list-item"
sc.states <- sc.url %>%
  html_nodes(sc.node) %>%
  html_text()
```

```
load("sc-geocode-data.RData")
# View the initial data
sc.states
```

```
##  [1] "\r\n        Puerto Rico\r\n    "
##  [2] "\r\n        Alabama\r\n    "
##  [3] "\r\n        Arizona\r\n    "
##  [4] "\r\n        Arkansas\r\n    "
##  [5] "\r\n        Colorado\r\n    "
##  [6] "\r\n        Delaware\r\n    "
##  [7] "\r\n        Florida\r\n    "
##  [8] "\r\n        Georgia\r\n    "
##  [9] "\r\n        Idaho\r\n    "
## [10] "\r\n        Illinois\r\n    "
## [11] "\r\n        Indiana\r\n    "
## [12] "\r\n        Iowa\r\n    "
## [13] "\r\n        Kansas\r\n    "
## [14] "\r\n        Kentucky\r\n    "
## [15] "\r\n        Louisiana\r\n    "
## [16] "\r\n        Michigan\r\n    "
## [17] "\r\n        Mississippi\r\n    "
## [18] "\r\n        Missouri\r\n    "
## [19] "\r\n        Montana\r\n    "
## [20] "\r\n        Nebraska\r\n    "
## [21] "\r\n        New Jersey\r\n    "
## [22] "\r\n        New York\r\n    "
## [23] "\r\n        North Carolina\r\n    "
## [24] "\r\n        North Dakota\r\n    "
## [25] "\r\n        Ohio\r\n    "
## [26] "\r\n        Oklahoma\r\n    "
## [27] "\r\n        Pennsylvania\r\n    "
## [28] "\r\n        South Carolina\r\n    "
## [29] "\r\n        South Dakota\r\n    "
## [30] "\r\n        Tennessee\r\n    "
## [31] "\r\n        Texas\r\n    "
## [32] "\r\n        Utah\r\n    "
## [33] "\r\n        Virginia\r\n    "
## [34] "\r\n        West Virginia\r\n    "
## [35] "\r\n        Wisconsin\r\n    "
## [36] "\r\n        Wyoming\r\n    "
## [37] "\r\n        Puerto Rico\r\n    "
## [38] "\r\n        Alabama\r\n    "
## [39] "\r\n        Arizona\r\n    "
## [40] "\r\n        Arkansas\r\n    "
## [41] "\r\n        Colorado\r\n    "
## [42] "\r\n        Delaware\r\n    "
## [43] "\r\n        Florida\r\n    "
## [44] "\r\n        Georgia\r\n    "
## [45] "\r\n        Idaho\r\n    "
## [46] "\r\n        Illinois\r\n    "
## [47] "\r\n        Indiana\r\n    "
## [48] "\r\n        Iowa\r\n    "
## [49] "\r\n        Kansas\r\n    "
## [50] "\r\n        Kentucky\r\n    "
## [51] "\r\n        Louisiana\r\n    "
## [52] "\r\n        Michigan\r\n    "
## [53] "\r\n        Mississippi\r\n    "
## [54] "\r\n        Missouri\r\n    "
## [55] "\r\n        Montana\r\n    "
## [56] "\r\n        Nebraska\r\n    "
## [57] "\r\n        New Jersey\r\n    "
## [58] "\r\n        New York\r\n    "
## [59] "\r\n        North Carolina\r\n    "
## [60] "\r\n        North Dakota\r\n    "
## [61] "\r\n        Ohio\r\n    "
## [62] "\r\n        Oklahoma\r\n    "
## [63] "\r\n        Pennsylvania\r\n    "
## [64] "\r\n        South Carolina\r\n    "
## [65] "\r\n        South Dakota\r\n    "
## [66] "\r\n        Tennessee\r\n    "
## [67] "\r\n        Texas\r\n    "
## [68] "\r\n        Utah\r\n    "
## [69] "\r\n        Virginia\r\n    "
## [70] "\r\n        West Virginia\r\n    "
## [71] "\r\n        Wisconsin\r\n    "
## [72] "\r\n        Wyoming\r\n    "
```

This is looking good so far but definitely needs cleaning. We will clean it easily with the following code:

```r
sc.states.clean <- sc.states %>%
  str_replace("\r","") %>% # begin removing new line tag
  str_replace("\n","") %>% # can be combined with \r
  trimws() %>% # remove extra white space
  unique() # remove duplicated states
# PR is removed only because openintro cannot convert it
# I chose to filter instead of using case_when to create
# the lower case pr value
sc.states.clean <- sc.states.clean %>%
  data.frame(stringsAsFactors = F) %>%
  filter(!. %in% "Puerto Rico") %>%
  mutate(Short = tolower(state2abbr(.)))

names(sc.states.clean) <- c("Long","Short")
```

```r
load("sc-geocode-data.RData")
head(sc.states.clean)
```

```
##       Long Short
## 1  Alabama    al
## 2  Arizona    az
## 3 Arkansas    ar
## 4 Colorado    co
## 5 Delaware    de
## 6  Florida    fl
```

Great! Already a pretty decent amount of work but it yields exactly what we need! Now for the next step in this scraping analysis - getting the cities.

Getting the Cities

```r
sc.cities <- data.frame(stringsAsFactors = F)
for(i in sc.states.clean$Short){
  # base URL
  sc.url <- "https://stores.shoecarnival.com/"
  # adding states to the URL
  sc.url.state <- read_html(paste0(sc.url,i))
  # css code to access city info
  sc.node <- ".map-list-item"
  temp.data <- sc.url.state %>%
    html_nodes(sc.node) %>%
    html_text()
  # the data is messy just as before
  # we clean it the same way
  temp.data <- temp.data %>%
    str_replace("\r","") %>%
    str_replace("\n","") %>%
    trimws() %>%
    unique()
  # adding the city using the Shoe Carnival format
  # removing any spaces from city names
  # simple prep for the URL
  temp.data <- temp.data %>%
    data.frame(city.full = .,stringsAsFactors = F) %>%
    mutate(city.url = tolower(
      str_replace(city.full," ","")
      ), state = i)

  # bring it altogether
  sc.cities <- rbind(sc.cities,temp.data)
}
```

Now we have the state and each city for those states. The next step is to create our final URL:

```r
sc.cities <- sc.cities %>%
  mutate(full.url = paste("https://stores.shoecarnival.com",
                          state,
                          city.url,
                          sep = "/")
         )
```

```r
load("sc-geocode-data.RData")
head(sc.cities)
```

```
##   city.full city.url state                                   full.url
## 1   Decatur  decatur    al  https://stores.shoecarnival.com/al/decatur
## 2    Dothan   dothan    al   https://stores.shoecarnival.com/al/dothan
## 3  Florence florence    al https://stores.shoecarnival.com/al/florence
## 4     Foley    foley    al    https://stores.shoecarnival.com/al/foley
## 5   Gadsden  gadsden    al  https://stores.shoecarnival.com/al/gadsden
## 6    Hoover   hoover    al   https://stores.shoecarnival.com/al/hoover
```

Looking good! Now, we need those addresses! We will need to loop through the newly created URL's yet again.

Getting the Addresses

```r
sc.address <- data.frame(stringsAsFactors = F)
for(i in sc.cities$full.url){
  sc.url <- read_html(i)
  # the css tag for address
  sc.node <- ".address"
  temp.data <- sc.url %>%
    html_nodes(sc.node) %>%
    html_text()
  # additional data cleaning
  temp.data <- temp.data %>%
    str_replace("\r","") %>%
    str_replace("\n","") %>%
    trimws() %>%
    unique() %>%
    data.frame(stringsAsFactors = F)

  # bring it altogether
  sc.address <- rbind(sc.address,temp.data)

}

sc.address <- sc.address %>%
  unique()
```

```r
load("sc-geocode-data.RData")
head(sc.address)
```

```
##                                                                       .
## 1         1682 Beltline Rd SW\r\n                       Decatur, AL 35603
## 2 6275 University Drive NW\r\n                       Huntsville, AL 35806
## 3     2750 Carl T Jones Rd\r\n                       Huntsville, AL 35802
## 4        354C Cox Creek Pkwy\r\n                       Florence, AL 35630
## 5       3500 Ross Clark Circle\r\n                       Dothan, AL 36303
## 8            2601 S McKenzie St\r\n                       Foley, AL 36535
```

They always say that the hardest part is getting the data. While that may be true, luckily for us, we now have all the data that we need to geocode.

Geocoding the Addresses

This is a relatively straightforward section. The code required to geocode is very minimal, yet powerful. Here we will use my google API key to access their geocoder.

```
# devtools::install_github("dkahle/ggmap") # run this to install ggmap
# library(ggmap)

# add your api key (must enable billing -
# 40k free geocodes per month)
register_google(key = "key_id")

# mutate_geocode appends the geocoded address
# to the dataframe
names(sc.address) <- "Address"
sc.geocodes <- sc.address %>%
  mutate_geocode(Address)
```

Geocoding in R is as simple as that. The drawback is the limitations that Google has now placed on the API. I once was able to geocode as much as I wanted whenever I wanted. BUT the credit each month does allow up to 40,000 locations to be geocoded and this is plenty for me but could be problematic for a much larger corporation.

Nevertheless, let's view this data again for completeness. Particularly, I want to examine the Evansville Shoe Carnival locations

```
load("sc-geocode-data.RData")
head(sc.geocodes %>%
       filter(grepl("Evansville, IN",Address)
              )
     )
```

```
##                                                            Address
## 1        4200 Hogue Rd\r\n                    Evansville, IN 47712
## 2 805 N Green River Rd\r\n                    Evansville, IN 47715
##         lon      lat
## 1 -87.62496 37.98022
## 2 -87.49040 37.98391
```

Excellent. The data appears to be correct and we can verify this with Google.
Typing the first set of coordinates into Google Maps gives us this:
https://www.google.com/maps/place/37%C2%B058'48.8%22N+87%C2%B037'29.9%22W/@37.9802211,-87.6255072,19z/data=!3m1!4b1!4m5!3m4!1s0x0:0x0!8m2!3d37.98022!4d-87.62496
and with the second set we get this:
https://www.google.com/maps/place/37%C2%B059'02.1%22N+87%C2%B029'25.4%22W/@37.9839142,-87.4925887,17z/data=!3m1!4b1!4m5!3m4!1s0x0:0x0!8m2!3d37.98391!4d-87.4904

The geocoder gives us a rooftop level of accuracy for each of the Shoe Carnival locations. The points may not be 100% exact but they are VERY close.

Wrap-Up with Mapping in R!

We have scraped the data and geocoded it. Now would be a great time to visualize some of the results. Luckily for us, we don't need to go too far! **ggmap** not only geocodes but it maps data as well. While there are packages that deal with shapefiles and layers in R, for this exercise we want to create two simple, but complete, maps. One with the points but another to show more of the distribution of the Shoe Carnival locations.

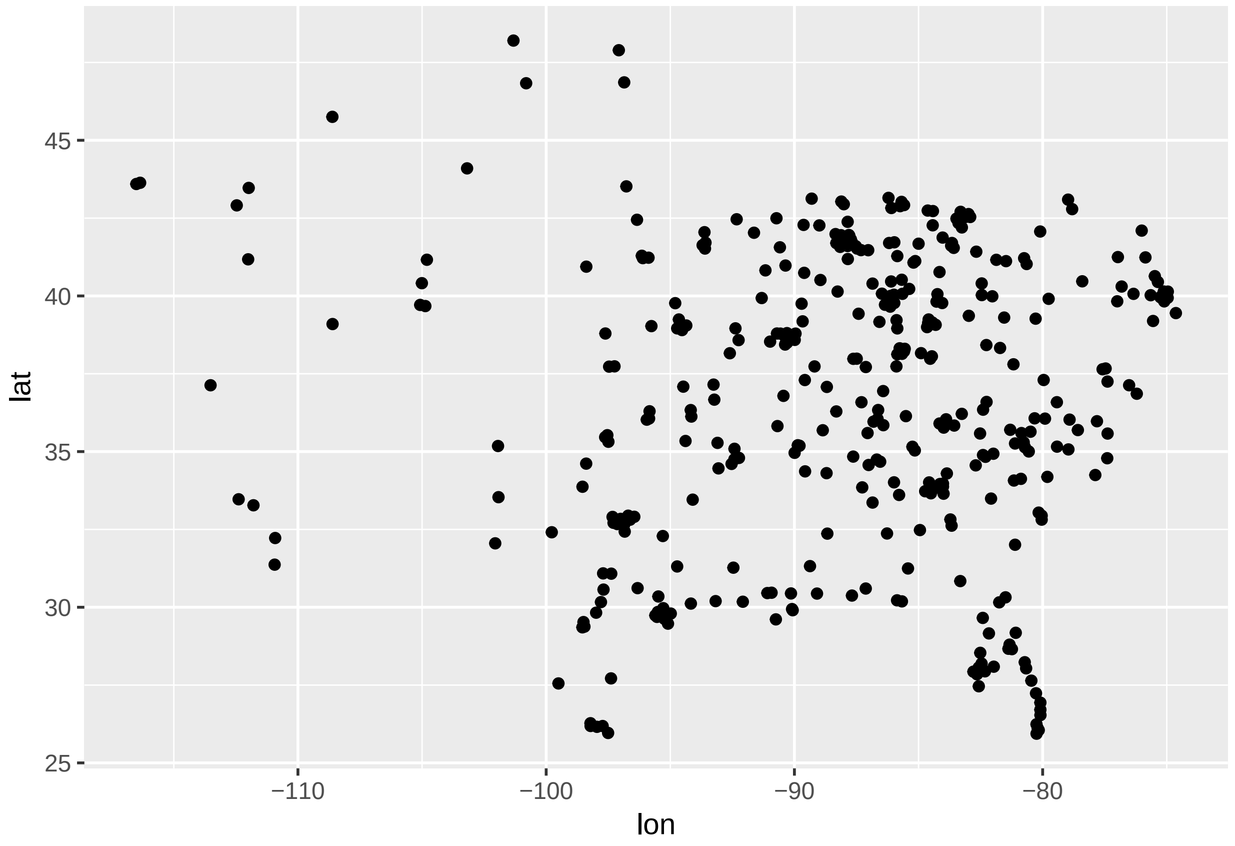
Let's start with the points map first:

```
# simple map with no layer
load("sc-geocode-data.RData")
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
sc.geocodes %>%
  ggplot(aes(x = lon,y = lat)) + geom_point()
```



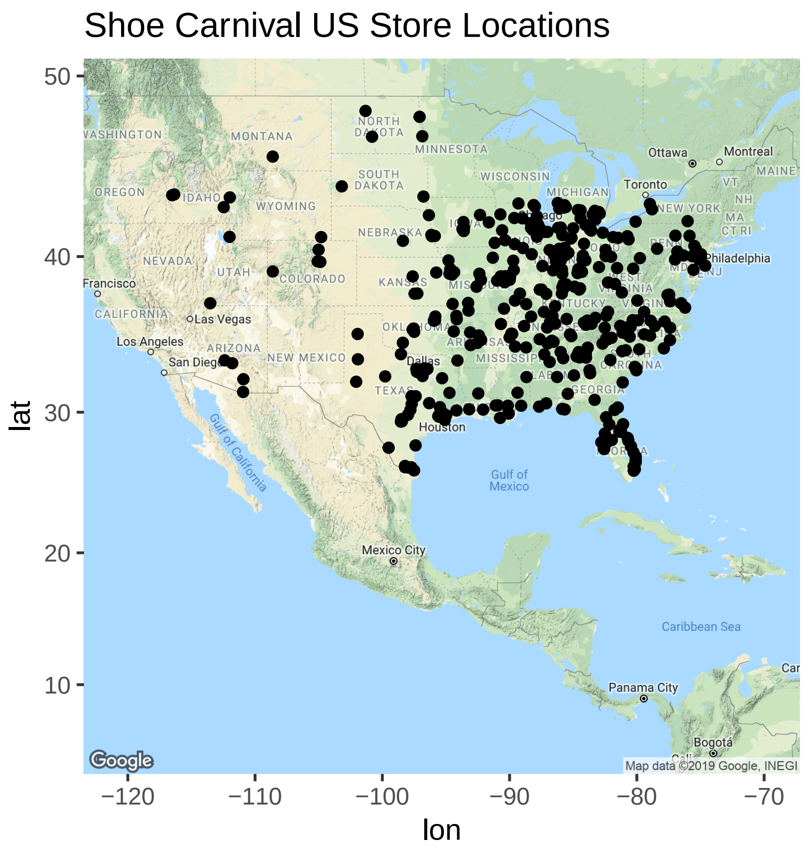
Yikes, that is pretty ugly. We can do better! Below we will create a Google Maps API Call.

```
# get a Google Maps layer of the US and adjust some of the styling
map <- get_googlemap(location="united states",
            source='google',
            zoom = 4,
            style=c(feature="administrative.country",element="labels",visibility="off")
            )
```

Now that we have a base layer for our map stored we can add the points and a title.

```
points.map.sc <- ggmap(map) +
  geom_point(data = sc.geocodes,aes(x = lon,y = lat)) +
  ggtitle("Shoe Carnival US Store Locations")
```

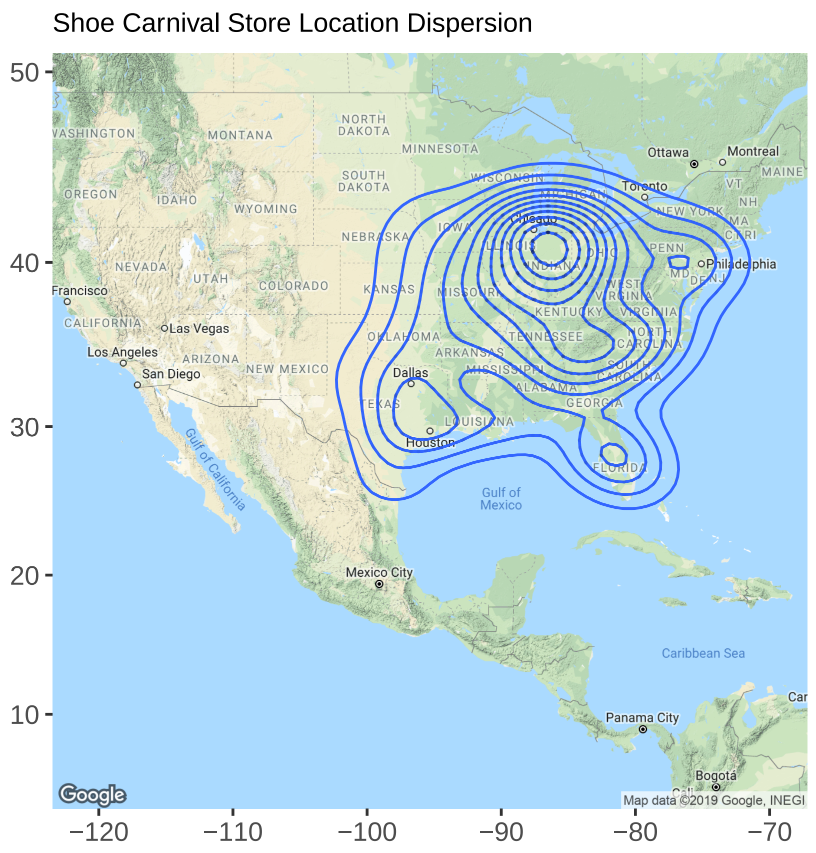
```
load("sc-geocode-data.RData")
points.map.sc
```



This looks significantly better! BUT can it be improved? There are quite a few points... Maybe we should look at the density of the map? Get an idea of how the Shoe Carnival locations are dispersed throughout the US. Of course we can tell that a majority appear to be on the Central/East Coast. What else can we see?

```
density.map.sc <- ggmap(map, extent = "panel", maprange=FALSE) +
  geom_density2d(data = sc.geocodes,aes(x = lon, y = lat)) +
  stat_density2d(data = sc.geocodes,aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
                 size = 0.01, geom = 'point') +
  scale_fill_gradient(low = "green", high = "red") +
  scale_alpha(range = c(0.00, 0.25), guide = FALSE) +
  theme(legend.position = "none", axis.title = element_blank(), text = element_text(size = 12))+
  ggtitle("Shoe Carnival Store Location Dispersion")+
  theme(plot.title = element_text(size=10))
```

```
load("sc-geocode-data.RData")
density.map.sc
```


Shoe Carnival Store Location Dispersion
50
40
30
20
10
−120
−110
−100
−90
−80
−70
Google
Map data ©2019 Google, INEGI

Beautiful. This really brings out the spread of the locations. This is only scratching the surface of the R GIS capabilities. But it is a very good start.

Taking Action

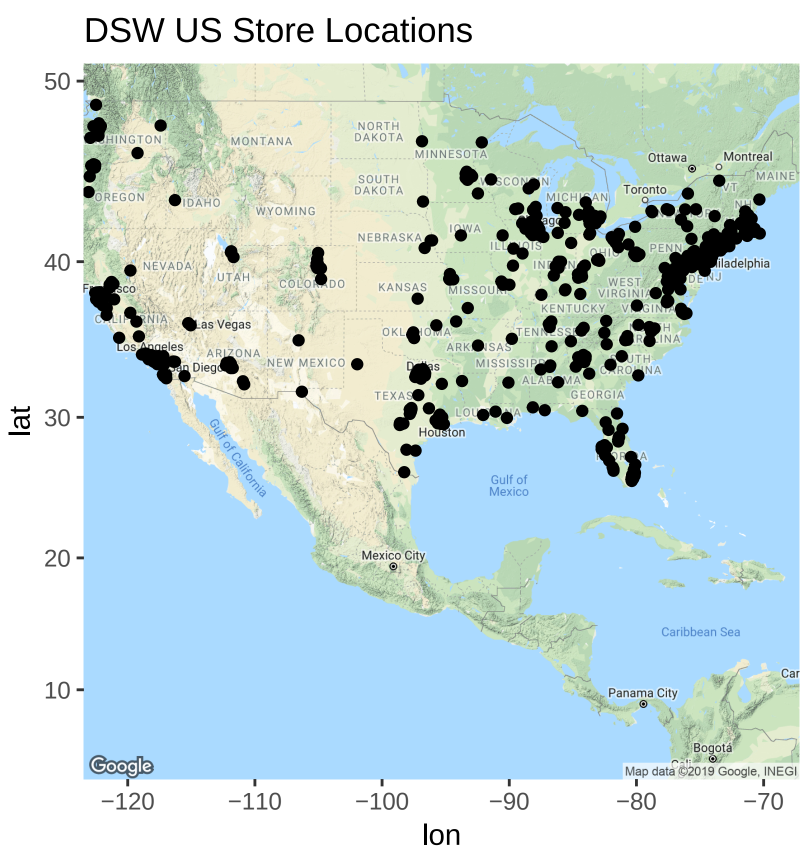
After working through all of this there is still the lingering question as to what can be done with this information. How does this lead to something actionable? Well, we know that Payless is closing doors. Are there locations that Shoe Carnival could target to try to gain some of the Payless consumer? What demographics does Shoe Carnival currently excel with and are there areas out west that could be targeted?

I did want to examine the Payless store locations but their website was not as easily accessible in regards to store locations. Luckily I was able to apply an almost identical scraping process to DSW stores. I will spare all of the scraping details but the final maps that I was able to produce using DSW points.

```r
points.map.dsw <- ggmap(map) +
  geom_point(data = dsw.geocodes,aes(x = lon,y = lat)) +
  ggtitle("DSW US Store Locations")
```

```r
load("sc-geocode-data.RData")
points.map.dsw
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```

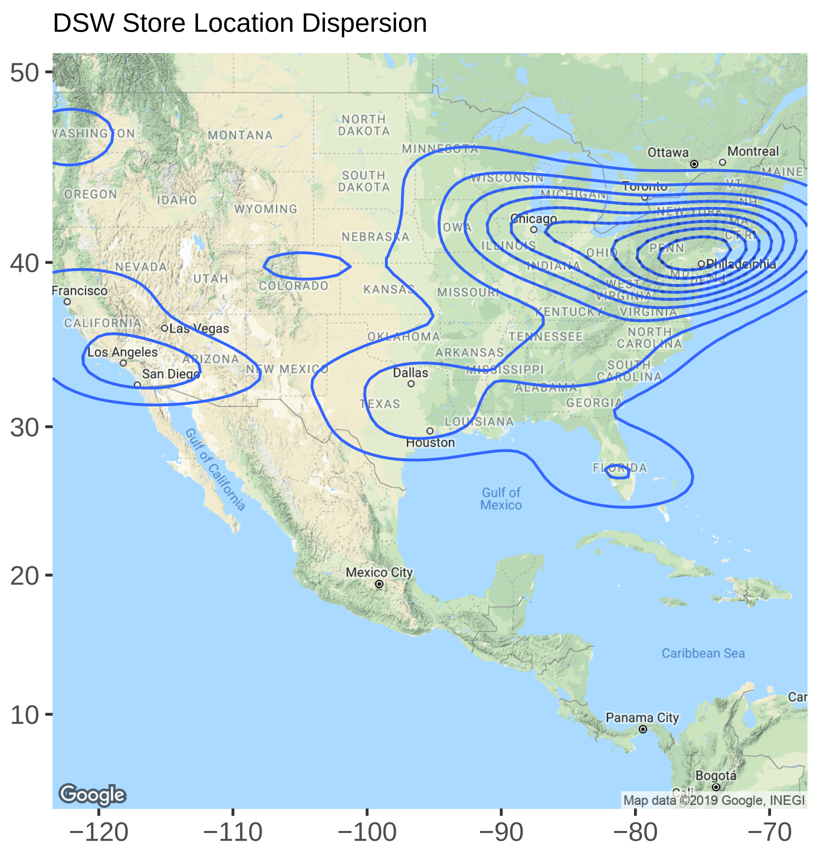


```
density.map.dsw <- ggmap(map, extent = "panel", maprange=FALSE) +
  geom_density2d(data = dsw.geocodes,aes(x = lon, y = lat)) +
  stat_density2d(data = dsw.geocodes,aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
                 size = 0.01, geom = 'point') +
  scale_fill_gradient(low = "green", high = "red") +
  scale_alpha(range = c(0.00, 0.25), guide = FALSE) +
  theme(legend.position = "none", axis.title = element_blank(), text = element_text(size = 12))+
  ggtitle("DSW Store Location Dispersion")+
  theme(plot.title = element_text(size=10))
```

```
load("sc-geocode-data.RData")
density.map.dsw
```

```
## Warning: Removed 5 rows containing non-finite values (stat_density2d).

## Warning: Removed 5 rows containing non-finite values (stat_density2d).
```



These maps alone tell a very different story than that of Shoe Carnival. DSW is heavily penetrated in the Northeast as well as a few hot spots on the West Coast. This is perhaps something worth considering in the future for store expansions. There does appear to be plenty of opportunity to hit high foot traffic and volume areas on the West Coast.

There are many different routes to take and other visuals that may help to better explore the data. This is certainly a great start and something to build off of in the future!

Thanks!

Thanks for working through this and I hope it has demonstrated some of the capabilities that I can bring to the company.

Citations

D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL **http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf**