**Load Libraries**

First, let’s load our favourite libraries.

|  |  |
| --- | --- |
| 1  2  3  4  5 | **library**(rvest)  **library**(readr)  **library**(tidyverse)  **library**(scales)  **library**(ggmap) |

**Figure out locations**

On its [site, Shake Shack](https://www.shakeshack.com/location) fortunately has all the locations and opening dates, going back to April 23, 2012. The archive pages run from 1 to 20 with this URL structure:

https://www.shakeshack.com/location/page/

Using [SelectorGadget](http://selectorgadget.com/), I figured out the XPath and CSS code to find the opening date, location name, and location page link. Then, I wrote a function to retrieve these values from a given archive page.

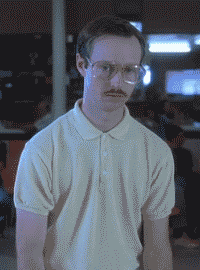
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | get\_locations <- **function**(**url**) {  page\_html <- read\_html(**url**)  nodes <- page\_html %>%  html\_nodes(xpath = '//\*[contains(concat( " ", @class, " " ), concat( " ", "span4", " " ))]')    **data.frame**(opdate = html\_nodes(x = nodes,  xpath = '//\*[contains(concat( " ", @class, " " ), concat( " ", "date", " " ))]') %>%  html\_text(trim = TRUE),  store\_loc\_name = html\_nodes(x = nodes,  css = 'h2') %>%  html\_text(trim = TRUE),  store\_loc\_link = html\_nodes(x = nodes,  css = 'h2 a') %>%  html\_attr("href"),  stringsAsFactors = FALSE)  } |

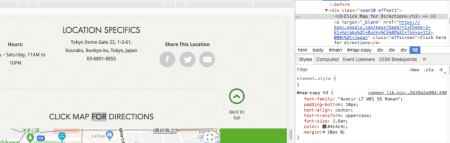
I applied this function to retrieve all location opening dates, names, and individual location urls:

|  |  |
| --- | --- |
| 1  2  3 | all\_loc\_pages <- paste0("https://www.shakeshack.com/location/page/", 1:20, "/")    all\_locations <- **do.call**(**rbind**, **lapply**(all\_loc\_pages, get\_locations)) |

**Find addresses of all locations**

If you visit an individual location’s page, such as [this Tokyo Dome page,](https://www.shakeshack.com/location/tokyo-japan-tokyo-dome) you will see that often the exact address is not listed, or if it is, you can’t directly geocode it. But, luckily, there’s a Google Map right below the location. I thought, they must be passing some parameters to Google Maps API. I spend a good amount of time, but couldn’t figure out how they were getting the map. And. Then. I found out that the text “CLICK MAP FOR DIRECTIONS” block had a valid address as part of the hyperlink!!





I wrote another simple function to get the addresses from the given URL:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | get\_loc\_cords <- **function**(loc\_url) {  location\_html <- read\_html(loc\_url)  **data.frame**(loc\_url = loc\_url,  goog\_map\_url = location\_html %>%  html\_nodes(xpath = '//a[text()="Click here for directions"]') %>%  html\_attr("href"),  stringsAsFactors = FALSE)  }    location\_google\_maps\_address <- **do.call**(**rbind**, **lapply**(all\_locations$store\_loc\_link, get\_loc\_cords)) |

Then I joined the location name with the address data frame:

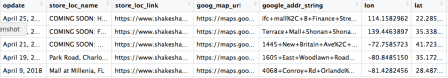
|  |  |
| --- | --- |
| 1 | all\_locations <- left\_join(all\_locations, location\_google\_maps\_address, **by** = **c**("store\_loc\_link" = "loc\_url")) |

**Geocoding the addresses**

Using the fantastic ggmap library and mutate\_geocode function, I geocoded all the addresses:

|  |  |
| --- | --- |
| 1  2  3 | all\_locations <- all\_locations %>%  mutate(google\_addr\_string = str\_sub(goog\_map\_url, **start** = 36)) %>%  mutate\_geocode(google\_addr\_string, output = "latlon") |

Here’s what the data frame looks like now:



**Tip**

You may want to create a Google developer key for mass geocoding. Since the mutate\_geocode function is used by many people, sometimes you may not get all the addresses geocoded. Use register\_google(key = , account\_type = 'premium', day\_limit = 100000) function to register your key with ggmap functions.

**Data manipulation**

Now that we have all the geographical coordinates, we just need to do some clean-up to get the data ready for plotting.

First, get the date field in order and add opening month and year columns:

|  |  |
| --- | --- |
| 1  2  3  4 | all\_locations <- all\_locations %>%  mutate(open\_date = **as.Date**(opdate, "%B %d, %Y"),  open\_month = lubridate::month(open\_date),  open\_year = lubridate::year(open\_date)) |

Second, get the cumulative count of store openings:

|  |  |
| --- | --- |
| 1  2  3  4  5 | ss\_op\_data\_smry <- all\_locations %>%  count(open\_date) %>%  ungroup() %>%  arrange(open\_date) %>%  mutate(cumm\_n = **cumsum**(n)) |

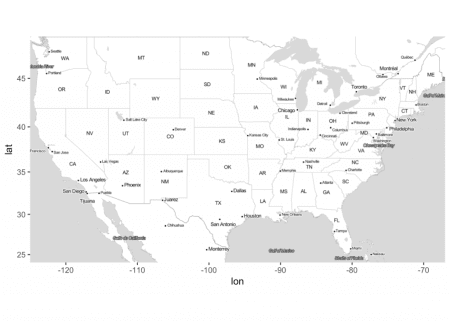
Third, join the summary back to the locations data frame:

|  |  |
| --- | --- |
| 1  2 | all\_locations\_smry <- inner\_join(all\_locations, ss\_op\_data\_smry,  **by** = **c**("open\_date" = "open\_date")) |

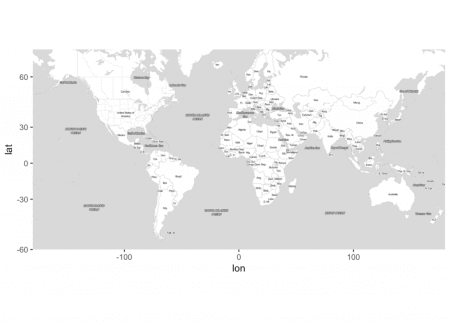
**Get the maps ready**

Using the ggmap library, I got the US map and a world map:

|  |  |
| --- | --- |
| 1  2 | us\_map <- get\_stamenmap(**c**(left = -125, bottom = 24, right = -67, top = 49), zoom = 5, maptype = "toner-lite")  ggmap(us\_map) |



|  |  |
| --- | --- |
| 1  2 | world\_map <- get\_stamenmap(bbox = **c**(left = -180, bottom = -60, right = 179.9999, top = 70), zoom = 3, maptype = "toner-lite")  ggmap(world\_map) |



**Create functions to plot each location**

Repurposing my code from the [Walmart spread](http://nandeshwar.info/data-visualization/walmart-growth-animation-r/) across the US, I wrote a similar function to plot locations with two different sizes: big, if the locations opened during the mapped month, and small, if the locations opened before the mapped month. I did so that we could notice the new locations.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | my\_us\_plot <- **function**(**df**, plotdate, mapid){  g <- ggmap(us\_map, darken = **c**("0.8", "black"), extent = "device")  old\_df <- **filter**(**df**, open\_date < plotdate)  new\_df <- **filter**(**df**, open\_date == plotdate)  # old locations  g <- g + geom\_point(**data** = old\_df, aes(x = lon, y = lat), size = 5, color = "dodgerblue", alpha = 0.4)  # new locations  g <- g + geom\_point(**data** = new\_df, aes(x = lon, y = lat), size = 8, color = "dodgerblue", alpha = 0.4)  g <- g + theme(axis.ticks = element\_blank(), axis.title = element\_blank(), axis.text = element\_blank(), plot.title = element\_blank(), panel.background = element\_rect(fill = "grey20"), plot.background = element\_rect(fill = "grey20"))  g <- g + annotate("text", x = -77, y = 33, label = "MONTH/YEAR:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -77, y = 32, label = paste0(**toupper**(**month.name**[**unique**(new\_df$open\_month)]), "/", **unique**(new\_df$open\_year)), color = "white", size = rel(6), fontface = 2, hjust = 0)  g <- g + annotate("text", x = -77, y = 31, label = "STORE COUNT:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -77, y = 30, label = comma(**unique**(new\_df$cumm\_n)), color = "white", size = rel(6), fontface = 2, hjust = 0)  filename <- paste0("maps/img\_" , str\_pad(mapid, 7, pad = "0"), ".png")  ggsave(filename = filename, **plot** = g, width = 13, height = 7, dpi = 120, type = "cairo-png")  } |

I modified this function to map the world:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | my\_world\_plot <- **function**(**df**, plotdate, mapid){  g <- ggmap(world\_map, darken = **c**("0.8", "black"), extent = "device")  old\_df <- **filter**(**df**, open\_date < plotdate)  new\_df <- **filter**(**df**, open\_date == plotdate)  g <- g + geom\_point(**data** = old\_df, aes(x = lon, y = lat), size = 5, color = "dodgerblue", alpha = 0.4)  g <- g + geom\_point(**data** = new\_df, aes(x = lon, y = lat), size = 8, color = "dodgerblue", alpha = 0.4)  g <- g + theme(axis.ticks = element\_blank(), axis.title = element\_blank(), axis.text = element\_blank(), plot.title = element\_blank(), panel.background = element\_rect(fill = "grey20"))  g <- g + annotate("text", x = -130, y = 0, label = "MONTH/YEAR:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -130, y = -10, label = paste0(**toupper**(**month.name**[**unique**(new\_df$open\_month)]), "/", **unique**(new\_df$open\_year)), color = "white", size = rel(6), fontface = 2, hjust = 0)  g <- g + annotate("text", x = -130, y = -20, label = "STORE COUNT:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -130, y = -30, label = comma(**unique**(new\_df$cumm\_n)), color = "white", size = rel(6), fontface = 2, hjust = 0)  filename <- paste0("maps/img\_" , str\_pad(mapid, 7, pad = "0"), ".png")  ggsave(filename = filename, **plot** = g, width = 12, height = 6, dpi = 150, type = "cairo-png")  } |

**Create maps**

Now, the exciting part: create month-by-month maps.

US maps:

|  |  |
| --- | --- |
| 1  2  3  4 | all\_locations\_smry %>%  mutate(mapid = group\_indices\_(all\_locations\_smry, .dots = 'open\_date')) %>%  group\_by(open\_date) %>%  do(pl = my\_us\_plot(all\_locations\_smry, **unique**(.$open\_date), **unique**(.$mapid))) |

World maps:

|  |  |
| --- | --- |
| 1  2  3  4 | all\_locations\_smry %>%  mutate(mapid = group\_indices\_(all\_locations\_smry, .dots = 'open\_date')) %>%  group\_by(open\_date) %>%  do(pl = my\_world\_plot(all\_locations\_smry, **unique**(.$open\_date), **unique**(.$mapid))) |

**Create a movie**

Using ffmpeg, we can put all the images together to create a movie:

|  |  |
| --- | --- |
| 1  2  3 | # works on a mac  makemovie\_cmd <- paste0("ffmpeg -framerate 8 -y -pattern\_type glob -i '", paste0(**getwd**(), "/maps/"), "\*.png'", " -c:v libx264 -pix\_fmt yuv420p '", paste0(**getwd**(), "/maps/"), "movie.mp4'")  **system**(makemovie\_cmd) |

We can use the convert function to create a gif:

|  |  |
| --- | --- |
| 1  2  3 | # https://askubuntu.com/a/43767  makegif\_cmd <- paste0("convert -delay 8 -loop 0 ", paste0(**getwd**(), "/maps/"), "\*.png ", "animated.gif") # loop 0 for forever looping  **system**(makegif\_cmd) |

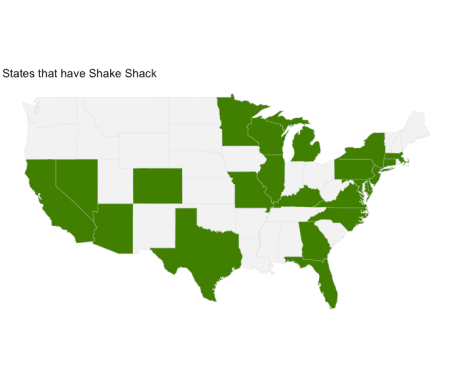
That’s it! We get nice looking videos showing location openings by each month. I was surprised to see how fast the company is opening the locations as well as how many locations it has in Asia!

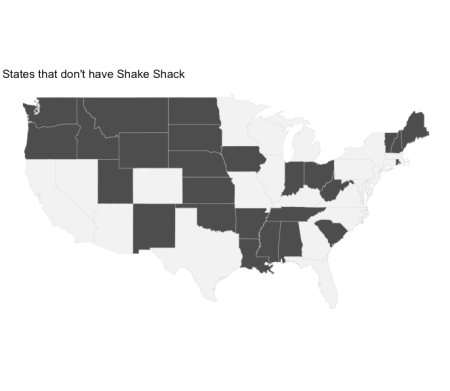
**Post hoc**

Using the ggimage library, I tried creating the maps using Shake Shack’s burger icon, but they didn’t turn out as good:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | my\_us\_icon\_plot <- **function**(**df**, plotdate, mapid){  g <- ggmap(us\_map, darken = **c**("0.8", "black"))  old\_df <- **filter**(**df**, open\_date < plotdate)  new\_df <- **filter**(**df**, open\_date == plotdate)  g <- g + geom\_image(**data** = old\_df, aes(x = lon, y = lat), **image** = "ss-app-logo.png", **by** = "height", size = 0.03, alpha = 0.4)  g <- g + geom\_image(**data** = new\_df, aes(x = lon, y = lat), **image** = "ss-app-logo.png", **by** = "height", size = 0.07, alpha = 0.4)  g <- g + theme(axis.ticks = element\_blank(), axis.title = element\_blank(), axis.text = element\_blank(), plot.title = element\_blank())  g <- g + annotate("text", x = -77, y = 33, label = "MONTH/YEAR:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -77, y = 32, label = paste0(**toupper**(**month.name**[**unique**(new\_df$open\_month)]), "/", **unique**(new\_df$open\_year)), color = "white", size = rel(6), fontface = 2, hjust = 0)  g <- g + annotate("text", x = -77, y = 31, label = "STORE COUNT:", color = "white", size = rel(5), hjust = 0)  g <- g + annotate("text", x = -77, y = 30, label = comma(**unique**(new\_df$cumm\_n)), color = "white", size = rel(6), fontface = 2, hjust = 0)  filename <- paste0("maps/img\_" , str\_pad(mapid, 7, pad = "0"), ".png")  ggsave(filename = filename, **plot** = g, width = 13, height = 7, dpi = 150, type = "cairo-png")  } |

**Fun maps**





What do you think? How else would you visualize these data points?