This icon map tutorial is going to extend our ggmap work to create icon maps with ggmap and ggimage.

**Set up packages**

Uncomment any “install.packages()” lines if you also need to install the package.

#install.packages("tidyverse")

library(tidyverse)

#install.packages("readr")

library(readr)

#install.packages("proj4")

library(proj4)

#install.packages("magick")

library(magick)

#install.packages("ggmap")

library(ggmap)

#install.packages("ggimage")

library(ggimage)

**Download the data**

I was able to find some neat location information on the [data portal for home town, London Ontario.](https://www.london.ca/city-hall/open-data/Pages/Open-Data-Data-Catalogue.aspx)

In their catalogue, they have a variety of spreadsheets with location information of public facilities. I downloaded all public facilities of interest, combined and standardized them.

#Download the data set

df= read\_csv('<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/MasterList.csv>', col\_names = TRUE)

#Set your API Key

#ggmap::register\_google(key = "ENTER\_YOUR\_API\_KEY")

**Convert UTM to lat and long**

In working with this dataset, I was very confused as to what the provided x and y coordinates were representing. After looking at their Q&A I realized that they are in Universal Transverse Mercator (UTM) format.

I found a way to convert UTM to Lat and Long. I slightly modified the code to run through the whole data set efficiently.

#Convert UTM to lat/lon

proj4string <- "+proj=utm +zone=17 +north +ellps=WGS84 +datum=NAD83 +units=m +no\_defs "

nRow <- nrow(df)

df$Lat <-0

df$Lon <-0

for(i in 1:nRow){

temp <-project(df[i,4:5], proj4string, inverse=TRUE)

df[i,6] <- temp$y

df[i,7] <-temp$x

}

**Create the starting map**

I start out with a basic map plotting all amenities, colored by facility type. Again, for more information on the basics of ggmap,

p <- ggmap(get\_googlemap(center = c(lon =-81.23304, lat = 42.98339),

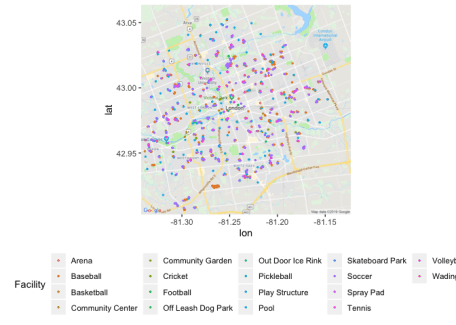
zoom = 12, scale = 2,

maptype ='roadmap',

color = 'color'))

p + geom\_point(aes(x = Lon, y = Lat, colour = Facility), data = df, size = 0.5) +

theme(legend.position="bottom")



**Filter and plot an icon map**

Filter to only include water facilities (pool, wading pool, spray pad).

Assign and display the icon for each facility type.

df2 <-df %>%

filter(Facility %in% c("Pool", "Wading Pool", "Spray Pad")) %>%

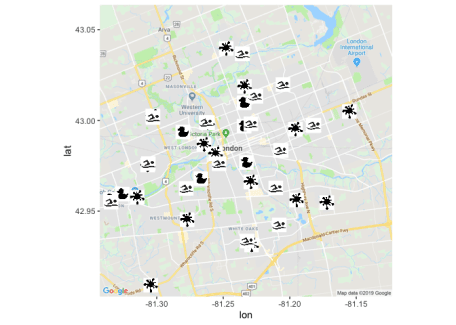
mutate(Image = case\_when(Facility == "Pool" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/swimmer.png>",

Facility == "Wading Pool" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/duckie.png>",

Facility == "Spray Pad" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/splash.png>"))

p + geom\_image(aes(x = Lon, y = Lat, image=Image), data = df2, size = 0.05) +

theme(legend.position="bottom")



**Prepare Images**

That was ok, but we should try to make the images more aesthetically pleasing using the magick package. We make each image transparent with the image\_transparent() function. We can also make the resulting image a specific color with image\_colorize().

I then saved the images using the image\_write() function. I manually re-uploaded them to GH.

baby <-image\_transparent(image\_read("<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/duckie.png>"), 'white')

baby <- image\_colorize(baby, 100, "#FF9933")

image\_write(baby, path = "babyPoolFinal.png", format = "png")

swimmer <-image\_transparent(image\_read("<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/swimmer.png>"), 'white')

swimmer <- image\_colorize(swimmer, 100, "#000099")

image\_write(swimmer, path = "swimmerFinal.png", format = "png")

splash <-image\_transparent(image\_read("<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/splash.png>"), 'white')

splash <- image\_colorize(splash, 100, "#3399FF")

image\_write(splash, path = "splashFinal.png", format = "png")

**Create a better map using the newly formatted icons**

df2 <-df %>%

filter(Facility %in% c("Pool", "Wading Pool", "Spray Pad")) %>%

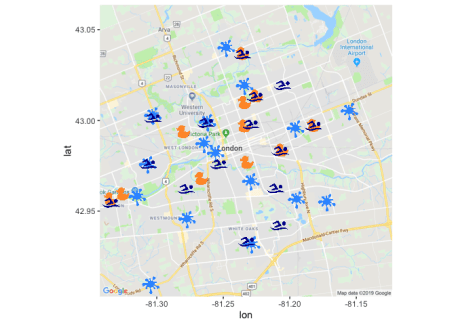
mutate(Image = case\_when(Facility == "Pool" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/swimmerFinal.png>",

Facility == "Wading Pool" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/babyPoolFinal.png>",

Facility == "Spray Pad" ~ "<https://raw.githubusercontent.com/lgellis/MiscTutorial/master/iconmap/splashFinal.png>"))

p + geom\_image(aes(x = Lon, y = Lat, image=Image), data = df2, size = 0.06) +

theme(legend.position="bottom")



**Create a new version using ‘stamen’ and ‘terrain-lines’**

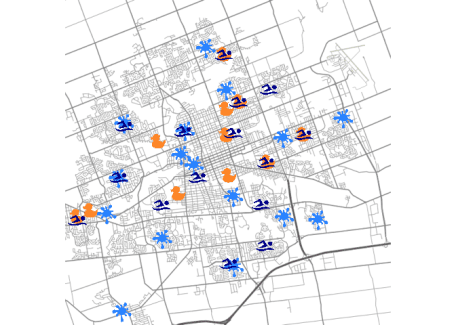
Now that we’ve sorted out our icons and have them plotted on the map, let’s try swapping out the base map. We are going to try the ‘stamen’ ‘terrain-lines’ version. I think that this a really cool way of looking at it as it represents the subdivisions more clearly.

center = c(lon =-81.23304, lat = 42.98339)

qmap(center, zoom = 12, source = "stamen", maptype = "terrain-lines") +

geom\_image(aes(x = Lon, y = Lat, image=Image), data = df2, size = 0.06) +

theme(legend.position="bottom")



**Create a new version using ‘stamen’ and ‘terrain’**

Now swap out ‘terrain-lines’ for ‘terrain’. I think this is my favorite version as it nicely balances showing green space and street lines very clearly at a distance.

center = c(lon =-81.23304, lat = 42.98339)

qmap(center, zoom = 12, source = "stamen", maptype = "terrain") +

geom\_image(aes(x = Lon, y = Lat, image=Image), data = df2, size = 0.06) +

theme(legend.position="bottom")

