**Finding nearby zipcodes within a certain radius using R**

Problem- you have zipcodes, but need to find nearby zipcodes within a certain radius. Let’s say you have a customer’s zipcode, and want to give them your nearby store locations that are reasonably far from them. Traditional +/- won’t work; what if living on the border of a state?

Solution- use the zip code API

1. Register for zip-code API key from https://www.zipcodeapi.com/
2. Specify your zip-code, radius, and unit of search within R

*zipcode <- 98105*

*distance <- 10*

*unit <- 'mile' #or 'km'*

*api\_key <- ‘key\_goes\_here’*

1. Form a string/ url using the zipcodeapi’s instructions-

https://www.zipcodeapi.com/API

My string looks like this-

*url <- paste0('www.zipcodeapi.com/rest/', api\_key, '/radius.csv/', zipcode, '/', distance, '/', unit)*

1. Query the zipcode API and get a resulting csv; query the API and read its reply into a dataframe using the following code:

*response <- GET(url)*

*zipcode.df <- read.csv(textConnection(content(response, 'text')))*

The resulting csv file is in the following format-

zip\_code, distance, city, state

98105, 0, “Seattle”, “WA”

98039, 4.586, "Medina", "WA"

…, …, …, …

1. You have a list of zipcodes now and want to find which of your shops are nearest to the customer zipcode. Assuming you have a dataframe of your shops where a column is zipcodes for your shop; you can create a subset/ dataset of your shop list that match the zipcodes sent back from the API through the following line-

*subset(shop\_dataframe, zipcode\_col %in% zipcode.df$zip\_code)*

It follows the following format where your shop dataframe is shop.df, the zipcode column within the shop.df is real.zip and the dataframe from the API is zipcode.df

subset(shop.df, real.zip %in% zipcode.df$zip\_code)

1. You know have a dataframe of shops closest to the customer! You can use this through directly showing the customer this information through text, or plot them visually on a map.

**Geocoding addresses/ adding coordinates to addresses**

Problem- standard ggmap only gives access to 50 queries a minute; 2500 a day; troubling if you have large dataset for which you need to add lat, long, address, etc. Need to geocode addresses- means- give an address to Google and get back location information to plot on a map. Relatively cheap at $0.50 USD / 1000; which means I’m paying $1.25 for my 5000 hospital dataset (minus 2,500 from free searches)

Solution- pay for Google Maps API key with 100,000 daily limit.

1. Get an unreleased version of ggmaps (v. 2.7 not installed on R) through

*devtools::install\_github("dkahle/ggmap")*

If you get errors, update R, RStudio, and install the missing packages it spits out.

1. Set your Google Maps API key if you're running it for the whole database; you can get one from the following link and set within R using

https://developers.google.com/maps/documentation/geocoding/usage-limits

*register\_google(key = “inser\_key\_here”)*

*register\_google(key = "key\_goes\_here", account\_type = "premium", day\_limit = 10000)*

1. Copy the for loop from here to add the lat, long, and address columns to your dataframe-

*for(i in 1:nrow(dataframe\_name)){*

*result <- geocode(dataframe $adrs\_col\_name[i], output = "latlona", source = "google")*

*dataframe\_name $lon[i] <- as.numeric(result[1])*

*dataframe\_name $lat[i] <- as.numeric(result[2])*

*dataframe\_name$geoAddress[i] <- as.character(result[3])*

*}*

1. Write a .csv file to save your data

*write.csv(dataframe\_name, "geocoded.csv", row.names=FALSE)*

1. Monitor usage from (may have to modify URL)

<https://console.developers.google.com/billing>

<https://developers.google.com/maps/documentation/geocoding/usage-limits>

<https://console.developers.google.com/billing/01D7B3-E697CA-F970EA/payment>

<https://console.developers.google.com/apis/dashboard?project=hospital-map-196904&duration=PT1H>

<https://console.developers.google.com/billing/unbilledinvoice?project=hospital-map-196904>

Note- set a modest daily limit that meets your budget; bearing in mind that at the time of writing this, each query was counted as three by Google and that it can take upto a second for each query; at an approximate rate of 16 minutes per 1000 queries.

Also note- the loop will occasionally stop when it cannot find information for an address; I suggest either manually or automatically within the function itself (not documented) print the row for this missing data and add it manually after the function has added it’s data. Do note that if so; the loop needs to be adjusted-

*for(i in start\_value:end\_value)){*

References/ Useful links:

<http://www.storybench.org/geocode-csv-addresses-r/>

**Making interactive map of data subset**