Today we’re going to use the [googleway](https://www.rdocumentation.org/packages/googleway/versions/2.7.1?tap_a=5644-dce66f&tap_s=10907-287229) R package, which allows their user to do requests to the GoogleMaps [Places API](https://developers.google.com/places/web-service/usage-and-billing). The goal is to create maps of specific places (restaurants, museums, etc.) with information from Google Maps rankings (number of stars given by other people). I already discussed this in french [here](http://nc233.com/2019/03/est-ce-que-cette-piscine-est-bien-notee/) to rank swimming pools in Paris. Let’s start by loading the three libraries I’m going to use : googleway, [leaflet](https://rstudio.github.io/leaflet/) to create animated maps, and RColorBrewer for color ranges.

library(googleway)

library(leaflet)

library(RColorBrewer)

First things first. To do API request to Google, we need an API key ; you can ask for one [here](https://developers.google.com/places/web-service/get-api-key). We’ll use this key for the rest of the program, so let’s declare a global variable :

api\_key <- "YourKeyHereIWontGiveYouMine"

**Retrieving Google Data**

We’re going to use the **google\_places** function to get a list of places matching a description, called *research* in my program (for instance : “Restaurant, Paris, France”). The output are multiple, and I’m going to store the place ID and the rating. I’ll also store the research token ; that’ll be explained later.

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key)

gmaps\_data <- gmaps\_request$results

place\_id <- gmaps\_data$place\_id

rating <- gmaps\_data$rating

token <- gmaps\_request$next\_page\_token

This function returns up to 20 places associated to the research by Google. If you want more than 20, you need to use the token previously stored in order to ask the Google Places API to give you the next results, by tweaking the function this way :

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key, page\_token = token)

There are tow caveats to this function. Firstly, the token can be NULL. In this case, there isn’t any further research results you can get. This happens automatically as soon as you reach 60 results. Secondly, the API needs time to refresh the token research (see [here](https://stackoverflow.com/questions/43942155/google-places-web-service-how-to-get-next-page-of-results-in-swift-3)) ; that’s why we’re going to make R wait a few seconds, using [Sys.sleep(time)](https://stat.ethz.ch/R-manual/R-devel/library/base/html/Sys.sleep.html) between our requests. Our complete function is therefore :

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key)

gmaps\_data <- gmaps\_request$results

place\_id <- gmaps\_data$place\_id

rating <- gmaps\_data$rating

token <- gmaps\_request$next\_page\_token

Sys.sleep(5)

continue <- TRUE

while (continue) {

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key, page\_token = token)

gmaps\_data <- gmaps\_request$results

if (!is.null(gmaps\_request$next\_page\_token)) {

place\_id = c(place\_id,gmaps\_data$place\_id)

rating = c(rating,gmaps\_data$rating)

token <- gmaps\_request$next\_page\_token

Sys.sleep(5)

}

else{continue <- FALSE}

}

Now we’re going to search for the spatial coordinates of the places we found. To this extent, we’re going to use the **google\_place\_details** function of the packages, and retrieve latitude and longitude with these two functions :

get\_lat <- function(id, key, language) {

id <- as.character(id)

details <- google\_place\_details(id, language = language, key=key)

return(details$result$geometry$location$lat)

}

get\_lng <- function(id, key, language) {

id <- as.character(id)

details <- google\_place\_details(id, language = language, key=key)

return(details$result$geometry$location$lng)

}

All these blocks add up to build the complete function :

get\_gmaps\_data <- function(research, api\_key, language) {

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key)

gmaps\_data <- gmaps\_request$results

place\_id <- gmaps\_data$place\_id

rating <- gmaps\_data$rating

token <- gmaps\_request$next\_page\_token

Sys.sleep(5)

continue <- TRUE

while (continue) {

gmaps\_request <- google\_places(search\_string = research, language = language, key = api\_key, page\_token = token)

gmaps\_data <- gmaps\_request$results

if (!is.null(gmaps\_request$next\_page\_token)) {

place\_id <- c(place\_id, gmaps\_data$place\_id)

rating <- c(rating, gmaps\_data$rating)

token <- gmaps\_request$next\_page\_token

Sys.sleep(5)

}

else{continue <- FALSE}

}

lat = sapply(place\_id, get\_lat, key=api\_key, language=language)

lng = sapply(place\_id, get\_lng, key=api\_key, language=language)

return(data.frame(place\_id, rating, lat, lng))

}

**Map plot**

The next part is more classical. We’re going to order the ratings of the data frame built by the previous function in order to arrange the places in differents groups. Each of the groups will be associated to a color on the data plot. If we want to make *number\_colors* groups with the color scheme *color* (for instance, “Greens”), we are using the following instructions :

color\_pal <- brewer.pal(number\_colors, color)

pal <- colorFactor(color\_pal, domain = seq(1,number\_colors))

plot\_data <- gmaps\_data

plot\_data$ranking <- ceiling(order(gmaps\_data$rating)\*number\_colors/nrow(plot\_data))

The definitive function just needs the addition of the leaflet call :

show\_map <- function(number\_colors, gmaps\_data, color="Greens") {

color\_pal <- brewer.pal(number\_colors,color)

pal <- colorFactor(color\_pal, domain = seq(1,number\_colors))

plot\_data <- gmaps\_data

plot\_data$ranking <- ceiling(order(gmaps\_data$rating)\*number\_colors/nrow(plot\_data)) leaflet(plot\_data) %>% addTiles() %>%

addCircleMarkers(

radius = 6,

fillColor = ~pal(ranking),

stroke = FALSE, fillOpacity = 1

) %>% addProviderTiles(providers$CartoDB.Positron)

}

**Examples**

I just need to combine these two functions in one, and then to do some food-related examples !

maps\_ranking\_from\_gmaps <- function(research, api\_key, language, number\_colors=5, color="Greens") {

show\_map(number\_colors, get\_gmaps\_data(research, api\_key, language), color)

}

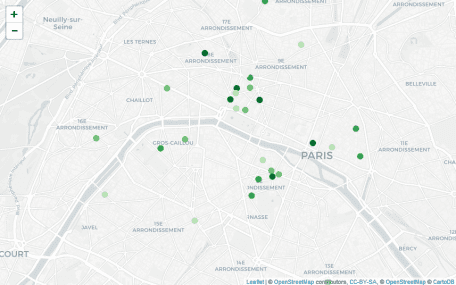
maps\_ranking\_from\_gmaps("Macaron, Paris, France", api\_key, "fr")

maps\_ranking\_from\_gmaps("Macaron, Montreal, Canada", api\_key, "fr")

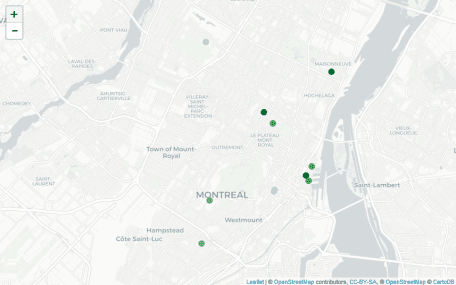
maps\_ranking\_from\_gmaps("Poutine, Montreal, Canada", api\_key, "fr", 5, "Blues")

maps\_ranking\_from\_gmaps("Poutine, Paris, France", api\_key, "fr", 5, "Blues")

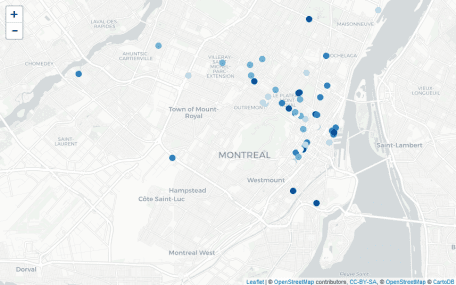
which returns the following maps :



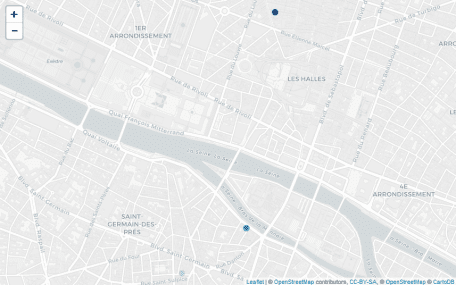
[Macaron](https://en.wikipedia.org/wiki/Macaron) in Paris, France



Macaron in Montreal, Canada



[Poutine](https://en.wikipedia.org/wiki/Poutine) in Montreal, Canada



Poutine in Paris, France (I guess French people are not ready for this)