Lab 7 Part 2

Exploring Google Maps Location Data

MIT 11.188/11.520

April 13, 2020

Rida Qadri

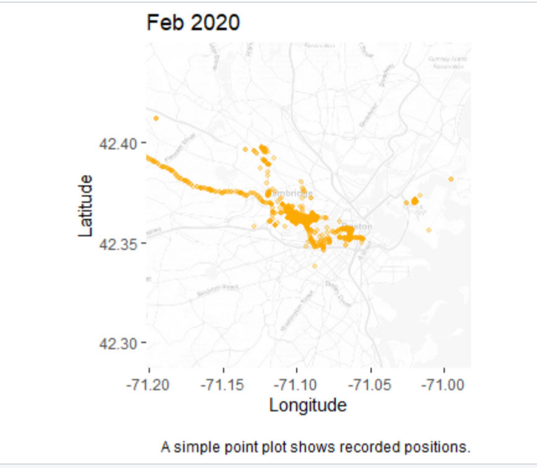
LAB 7 PROBLEM STATEMENT

Understand the food choices of one individual during lockdown, who is currently practicing social distancing in Central Square since March 12, 2020.

Part 1: Scraping Yelp Data using R

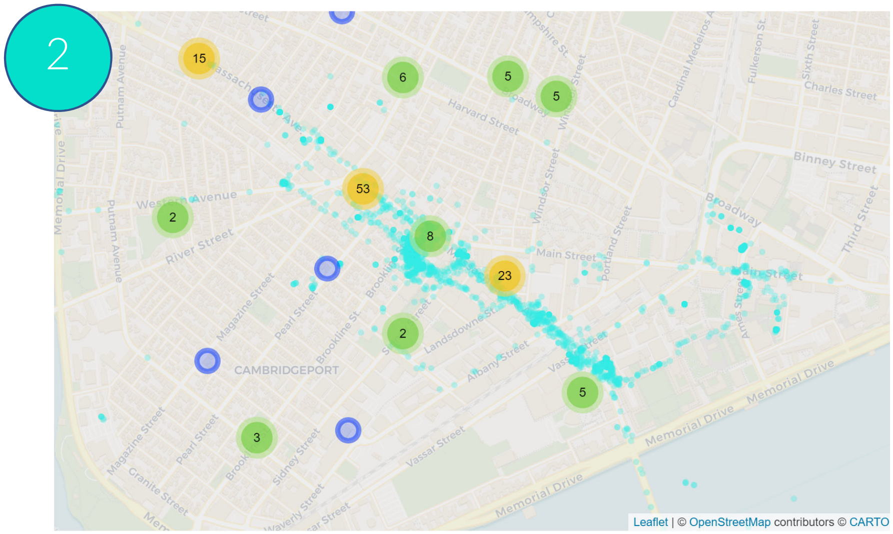
Part 2: Exploring Google Maps Location Data

1



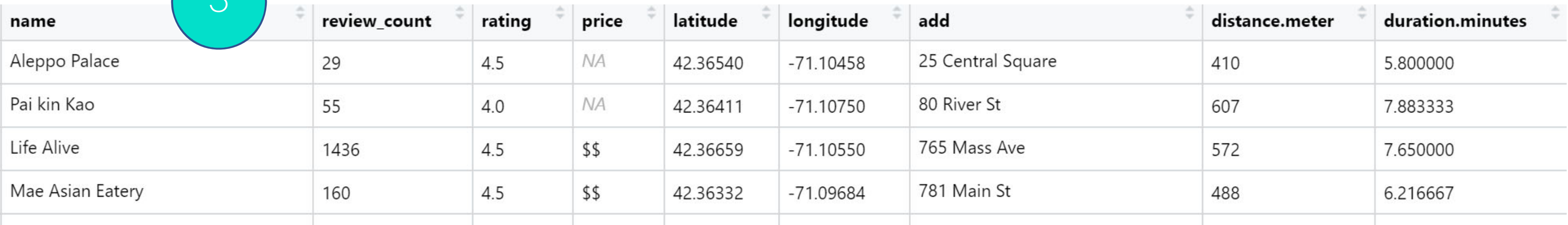
Extract Google Location history

2



Make an interactive map

3



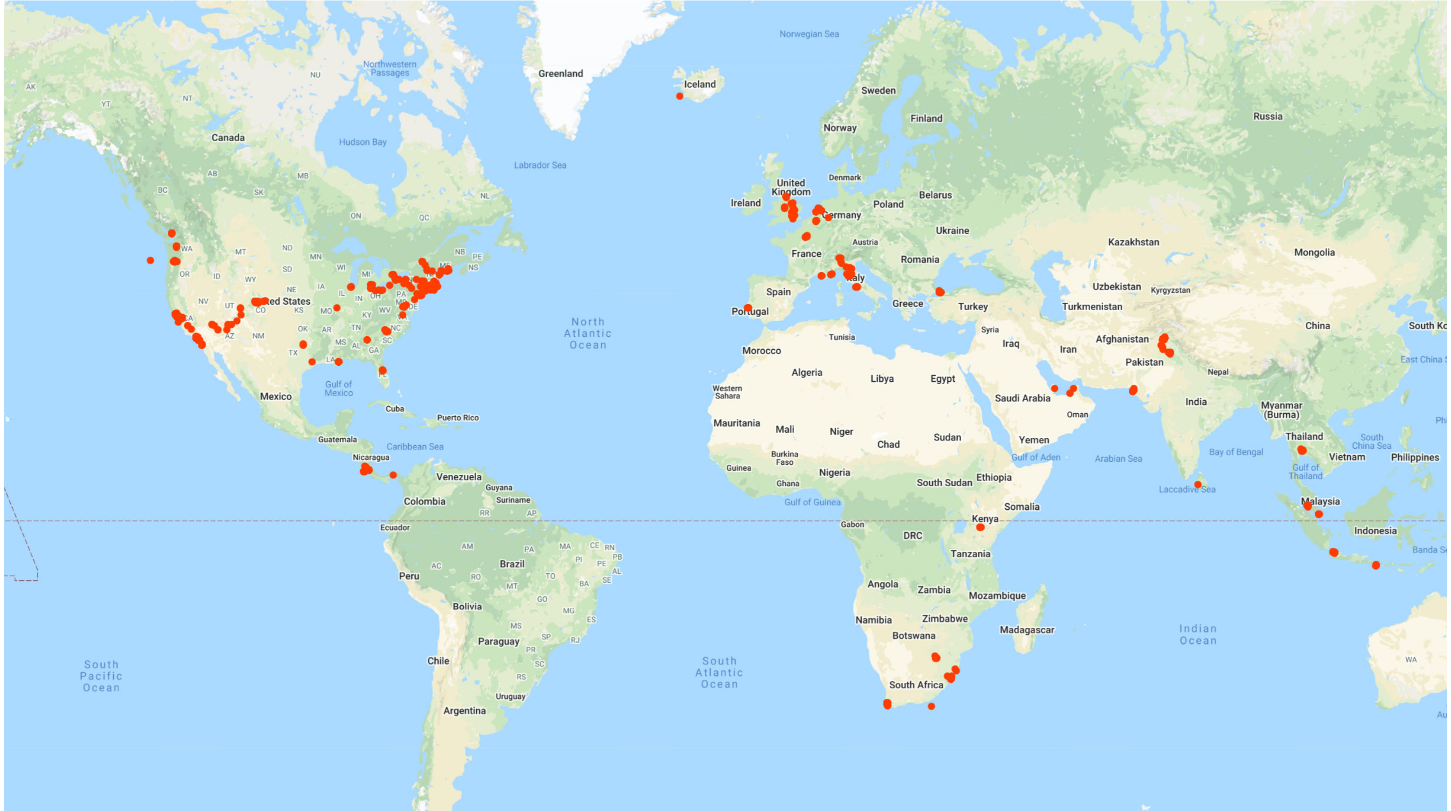
name	review_count	rating	price	latitude	longitude	add	distance.meter	duration.minutes
Aleppo Palace	29	4.5	NA	42.36540	-71.10458	25 Central Square	410	5.800000
Pai kin Kao	55	4.0	NA	42.36411	-71.10750	80 River St	607	7.883333
Life Alive	1436	4.5	$$	42.36659	-71.10550	765 Mass Ave	572	7.650000
Mae Asian Eatery	160	4.5	$$	42.36332	-71.09684	781 Main St	488	6.216667

Calculate walking distance to each restaurant

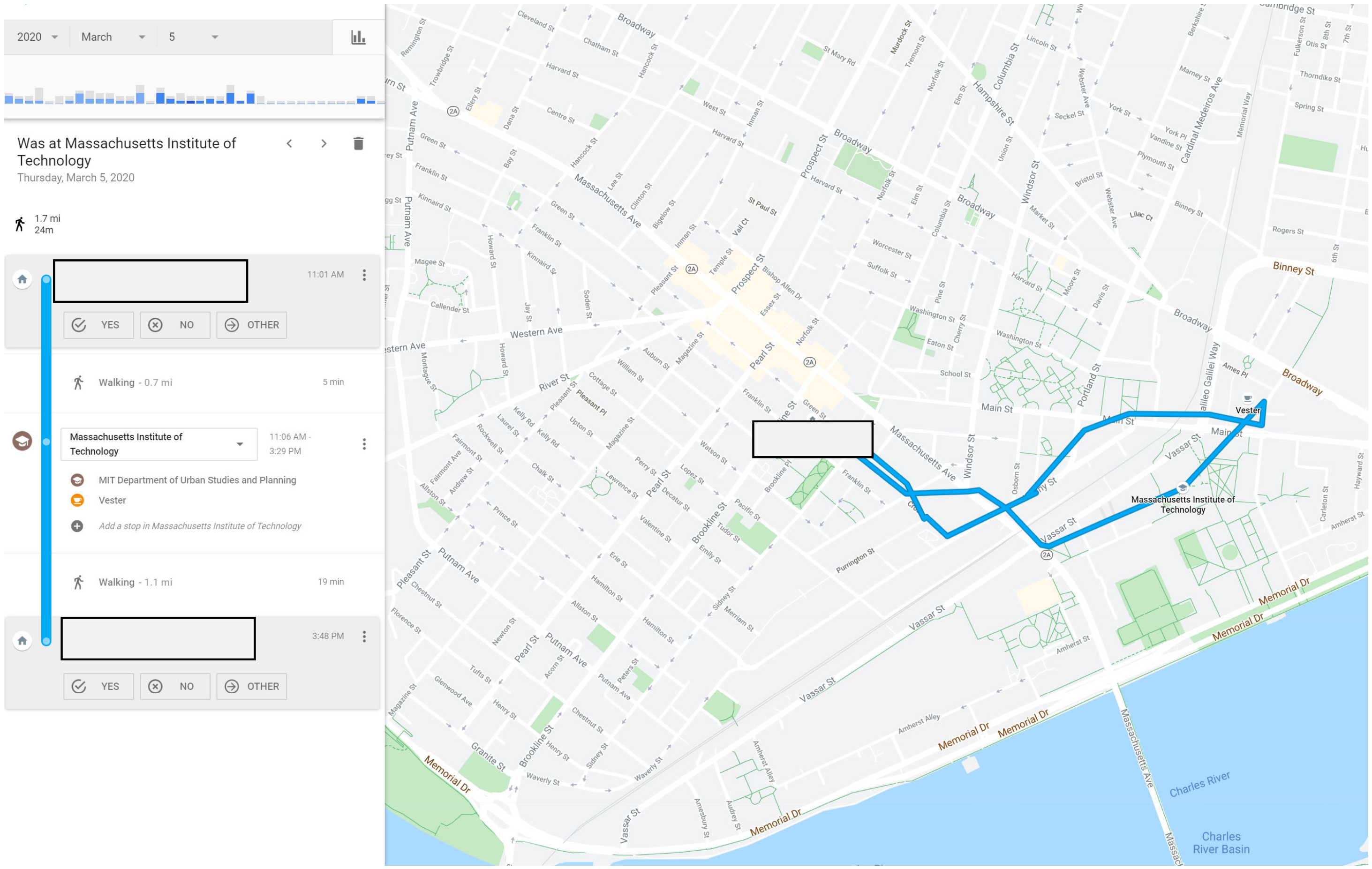
WALKING DISTANCE TO EACH RESTAURANT

- We have locations of all food options in a 1000m radius
- How do we get data on where this person lives?
- How do we get data on walking distance to each restaurant?

GOOGLE LOCATION HISTORY DATA



GOOGLE LOCATION HISTORY DATA



2020 ▾ March ▾ 5 ▾

Was at Massachusetts Institute of Technology

Thursday, March 5, 2020

1.7 mi
24m

[redacted] 11:01 AM

YES NO OTHER

Walking - 0.7 mi 5 min

Massachusetts Institute of Technology ▾ 11:06 AM - 3:29 PM

MIT Department of Urban Studies and Planning

Vester

Add a stop in Massachusetts Institute of Technology

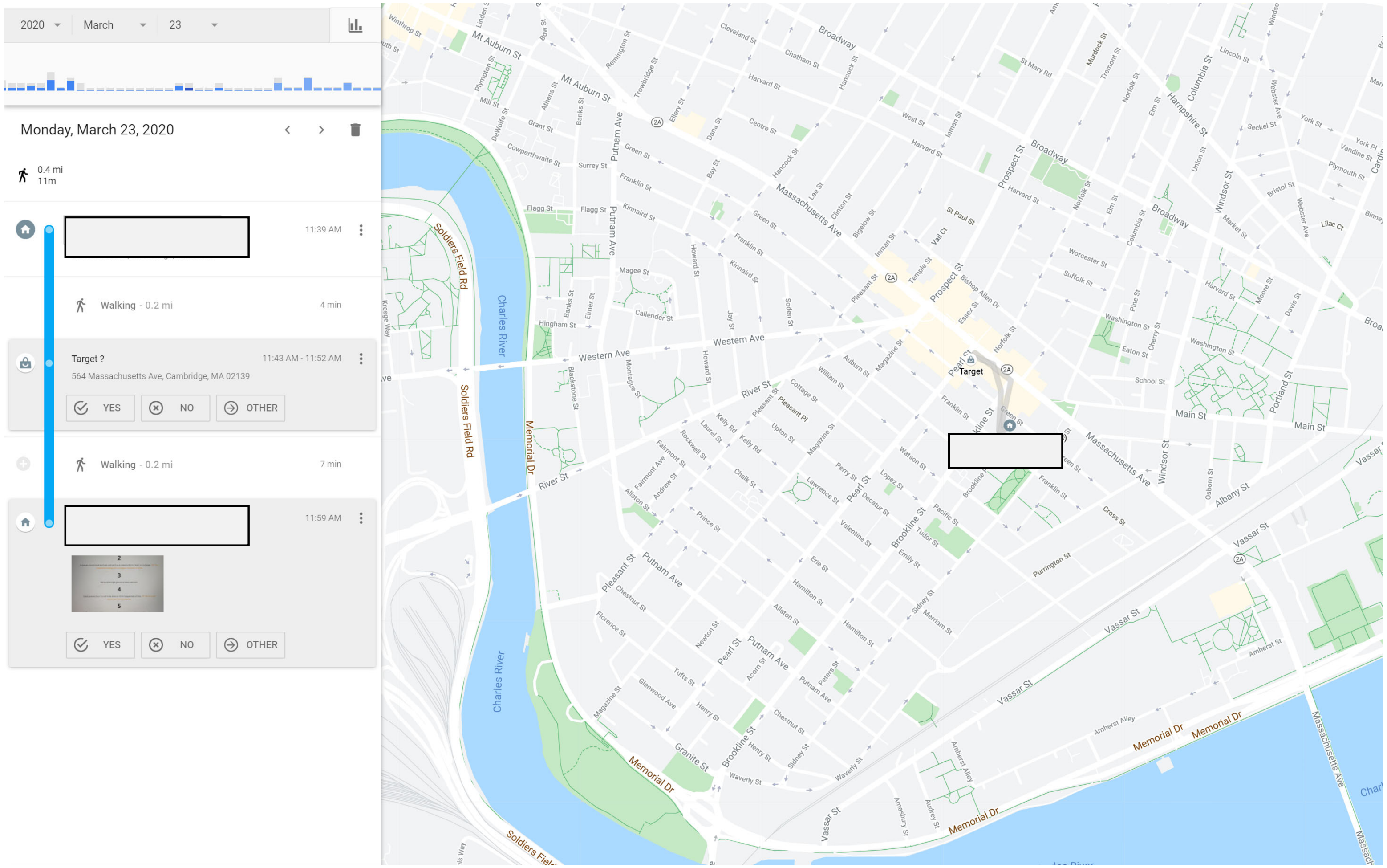
Walking - 1.1 mi 19 min

[redacted] 3:48 PM

YES NO OTHER

[redacted]

GOOGLE LOCATION HISTORY DATA



2020 ▾ March ▾ 23 ▾

Monday, March 23, 2020

0.4 mi
11m

[redacted] 11:39 AM

Walking - 0.2 mi 4 min

Target ? 11:43 AM - 11:52 AM

564 Massachusetts Ave, Cambridge, MA 02139

YES NO OTHER

Walking - 0.2 mi 7 min

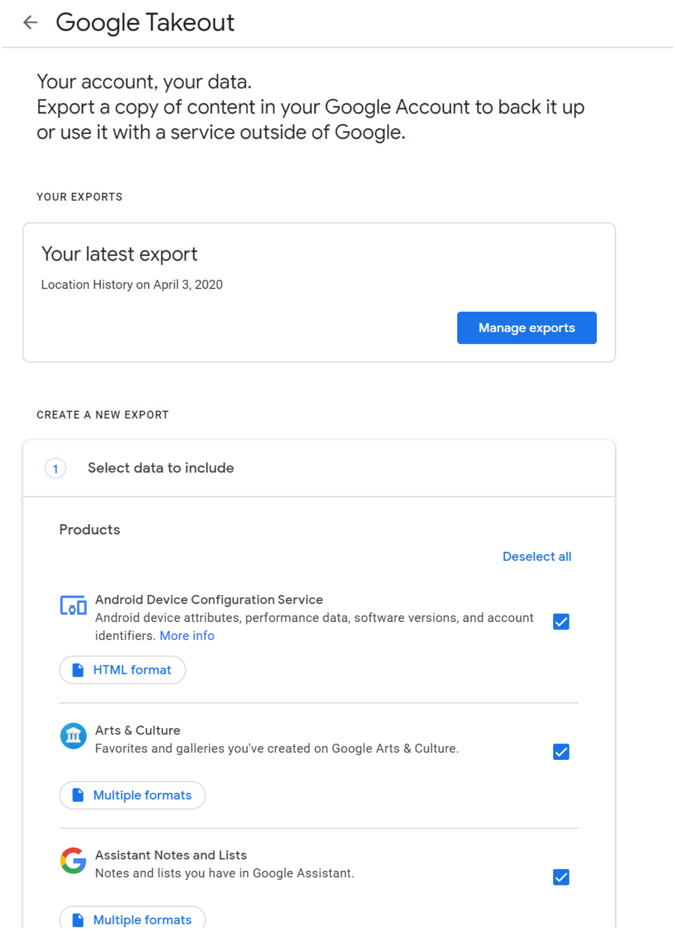
[redacted] 11:59 AM

YES NO OTHER

[redacted]

HOW DO WE ACCESS THIS DATA?

Takeout.google.com



← Google Takeout

Your account, your data.
Export a copy of content in your Google Account to back it up or use it with a service outside of Google.

YOUR EXPORTS

Your latest export

Location History on April 3, 2020

CREATE A NEW EXPORT

1 Select data to include

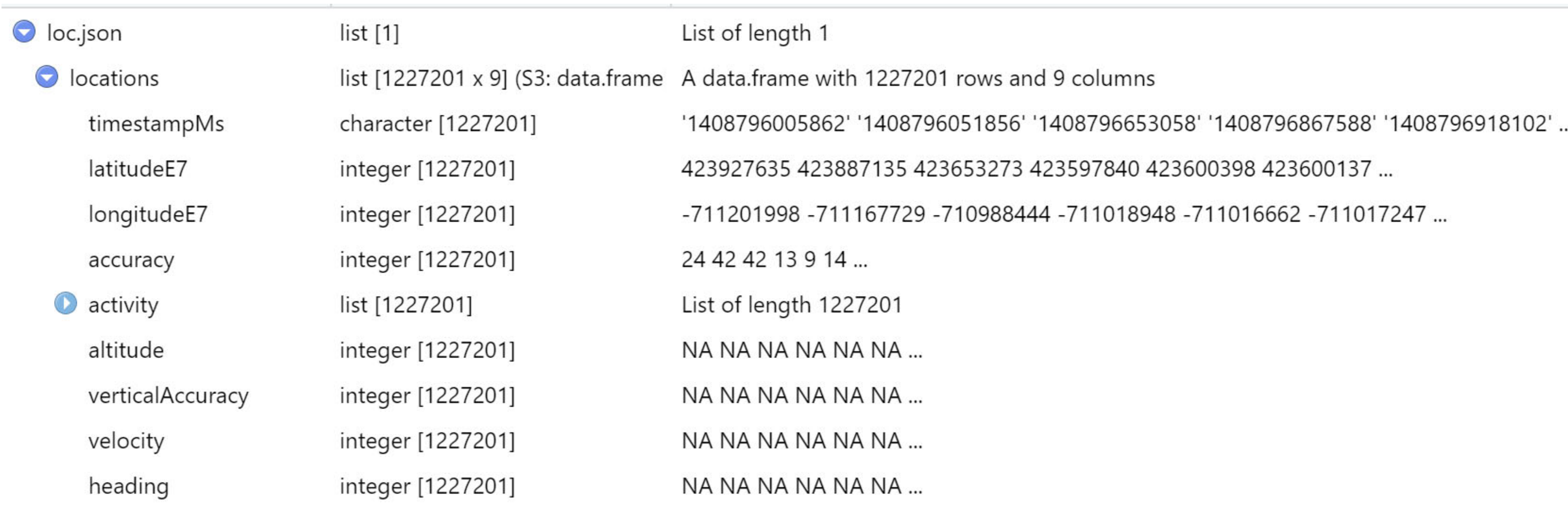
Products

Android Device Configuration Service ☑
Android device attributes, performance data, software versions, and account identifiers. More info

Arts & Culture ☑
Favorites and galleries you've created on Google Arts & Culture.

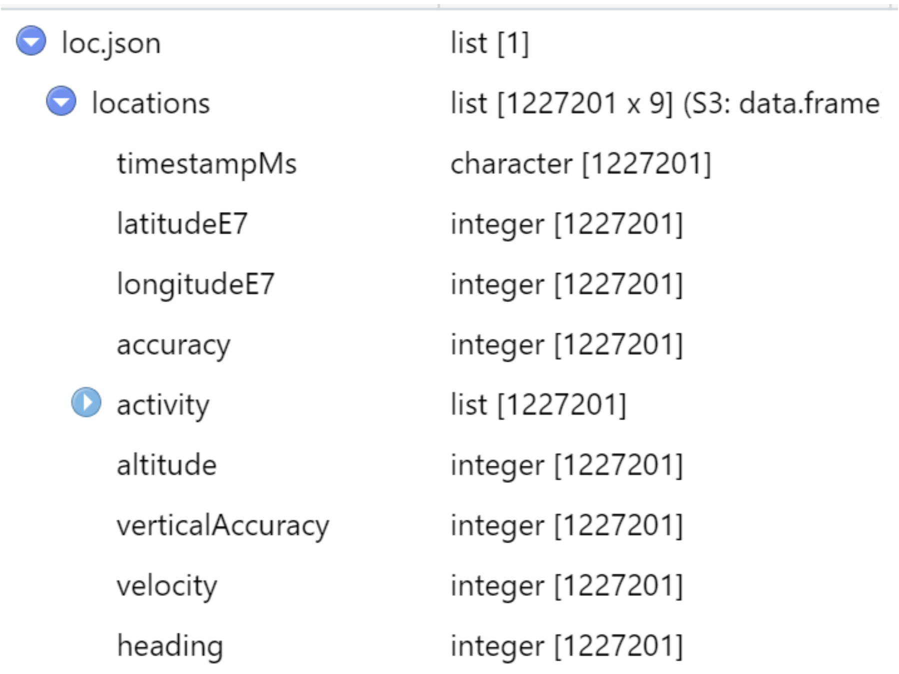
Assistant Notes and Lists ☑
Notes and lists you have in Google Assistant.

Access and Clean JSON File



loc.json	list [1]	List of length 1
locations	list [1227201 x 9] (S3: data.frame	A data.frame with 1227201 rows and 9 columns
timestampMs	character [1227201]	'1408796005862' '1408796051856' '1408796653058' '1408796867588' '1408796918102' ..
latitudeE7	integer [1227201]	423927635 423887135 423653273 423597840 423600398 423600137 ...
longitudeE7	integer [1227201]	-711201998 -711167729 -710988444 -711018948 -711016662 -711017247 ...
accuracy	integer [1227201]	24 42 42 13 9 14 ...
activity	list [1227201]	List of length 1227201
altitude	integer [1227201]	NA NA NA NA NA NA ...
verticalAccuracy	integer [1227201]	NA NA NA NA NA NA ...
velocity	integer [1227201]	NA NA NA NA NA NA ...
heading	integer [1227201]	NA NA NA NA NA NA ...

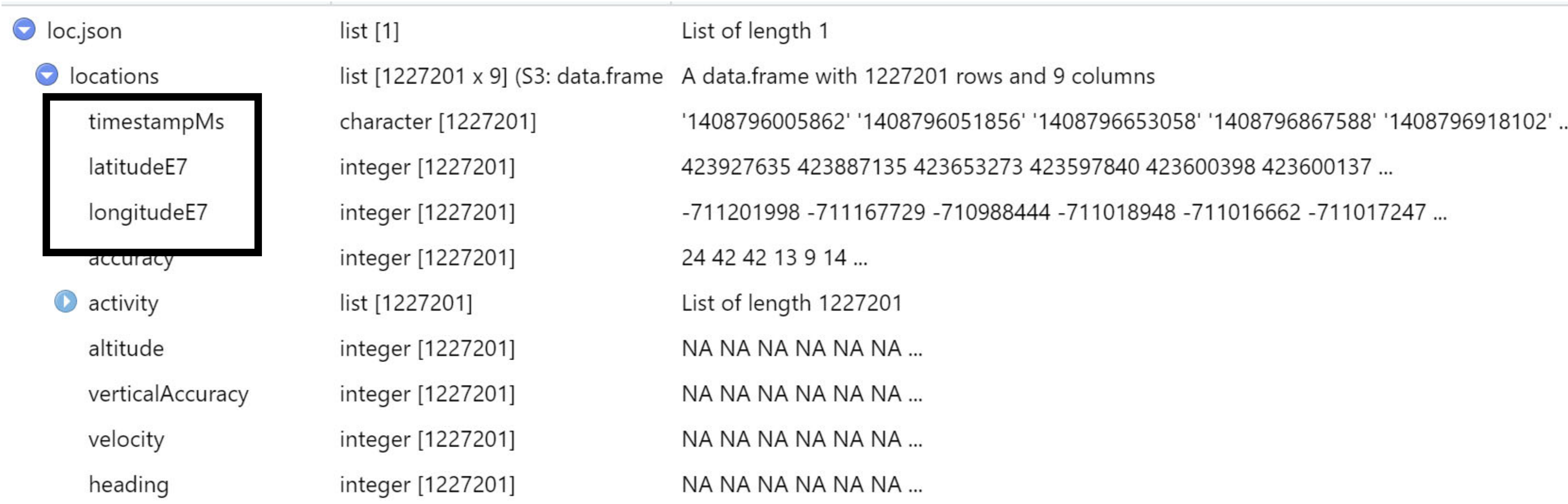
Access and Clean JSON File



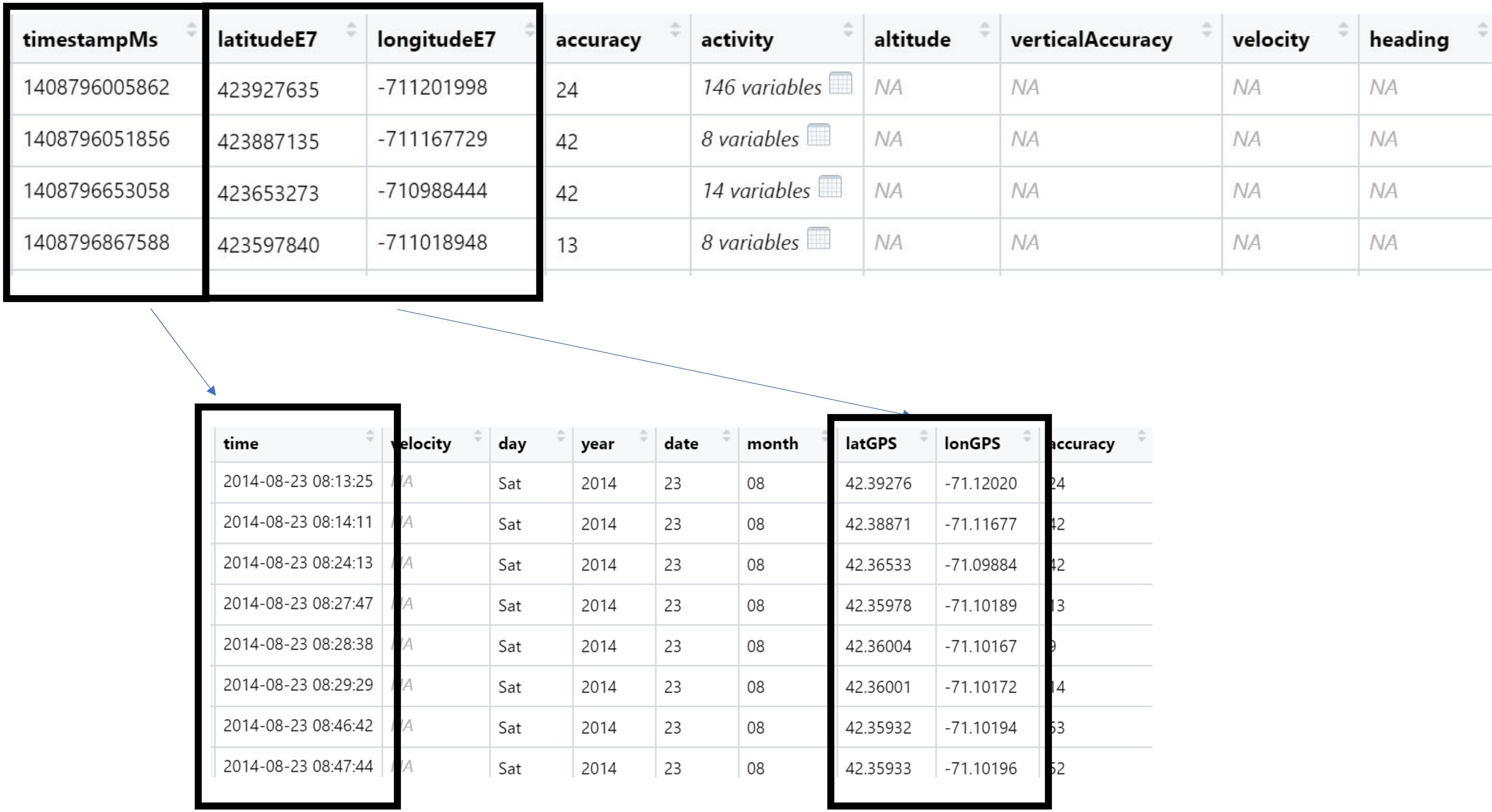
loc.json	list [1]
locations	list [1227201 x 9] (S3: data.frame
timestampMs	character [1227201]
latitudeE7	integer [1227201]
longitudeE7	integer [1227201]
accuracy	integer [1227201]
activity	list [1227201]
altitude	integer [1227201]
verticalAccuracy	integer [1227201]
velocity	integer [1227201]
heading	integer [1227201]

- timestampMs (int64): Timestamp (UTC) in milliseconds for the recorded location.
- latitudeE7 (int32): The latitude value of the location in E7 format (degrees multiplied by 10**7 and rounded to the nearest integer).
- longitudeE7 (int32): The longitude value of the location in E7 format (degrees multiplied by 10**7 and rounded to the nearest integer).
- Accuracy (int32): Approximate location accuracy radius in meters.
- Velocity (int32): Speed in meters per second.
- Heading (int32): Degrees east of true north.
- Altitude (int32): Meters above the WGS84 reference ellipsoid.
- verticalAccuracy (int32): Vertical accuracy calculated in meters.
- activity: Information about the activity at the location.
- timestampMs (int64): Timestamp (UTC) in milliseconds for when the datapoint was recorded
- type: Description of the activity type.
- Confidence (int32): Confidence associated with the specified activity type.

Access and Clean JSON File



loc.json	list [1]	List of length 1
locations	list [1227201 x 9] (S3: data.frame	A data.frame with 1227201 rows and 9 columns
timestampMs	character [1227201]	'1408796005862' '1408796051856' '1408796653058' '1408796867588' '1408796918102' ..
latitudeE7	integer [1227201]	423927635 423887135 423653273 423597840 423600398 423600137 ...
longitudeE7	integer [1227201]	-711201998 -711167729 -710988444 -711018948 -711016662 -711017247 ...
accuracy	integer [1227201]	24 42 42 13 9 14 ...
activity	list [1227201]	List of length 1227201
altitude	integer [1227201]	NA NA NA NA NA NA ...
verticalAccuracy	integer [1227201]	NA NA NA NA NA NA ...
velocity	integer [1227201]	NA NA NA NA NA NA ...
heading	integer [1227201]	NA NA NA NA NA NA ...



timestampMs	latitudeE7	longitudeE7	accuracy	activity	altitude	verticalAccuracy	velocity	heading
1408796005862	423927635	-711201998	24	*146 variables* ▦	*NA*	*NA*	*NA*	*NA*
1408796051856	423887135	-711167729	42	*8 variables* ▦	*NA*	*NA*	*NA*	*NA*
1408796653058	423653273	-710988444	42	*14 variables* ▦	*NA*	*NA*	*NA*	*NA*
1408796867588	423597840	-711018948	13	*8 variables* ▦	*NA*	*NA*	*NA*	*NA*

time	velocity	day	year	date	month	latGPS	lonGPS	accuracy
2014-08-23 08:13:25	*NA*	Sat	2014	23	08	42.39276	-71.12020	24
2014-08-23 08:14:11	*NA*	Sat	2014	23	08	42.38871	-71.11677	42
2014-08-23 08:24:13	*NA*	Sat	2014	23	08	42.36533	-71.09884	42
2014-08-23 08:27:47	*NA*	Sat	2014	23	08	42.35978	-71.10189	13
2014-08-23 08:28:38	*NA*	Sat	2014	23	08	42.36004	-71.10167	9
2014-08-23 08:29:29	*NA*	Sat	2014	23	08	42.36001	-71.10172	14
2014-08-23 08:46:42	*NA*	Sat	2014	23	08	42.35932	-71.10194	53
2014-08-23 08:47:44	*NA*	Sat	2014	23	08	42.35933	-71.10196	52

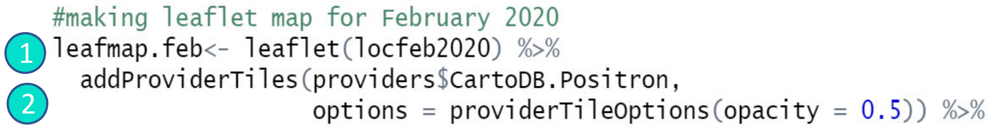
INTERACTIVE MAPS WITH LEAFLET

- Leaflet library
  - Map ‘widget’
  - Dataset
  - Basemap
  - Markers/symbols/Layers
  - ‘Frame’ or extent of map

Let’s see the kinds of maps we’re talking about

WHAT’S HAPPENING IN THE CODE

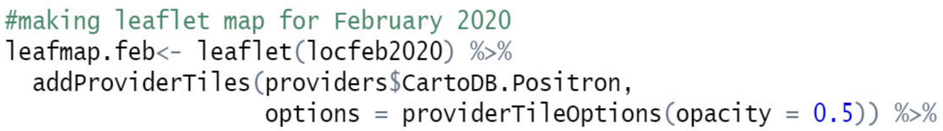
1. Calling ‘leaflet’ function which initializes a map widget storing data from locfeb2020
2. Defining basemap



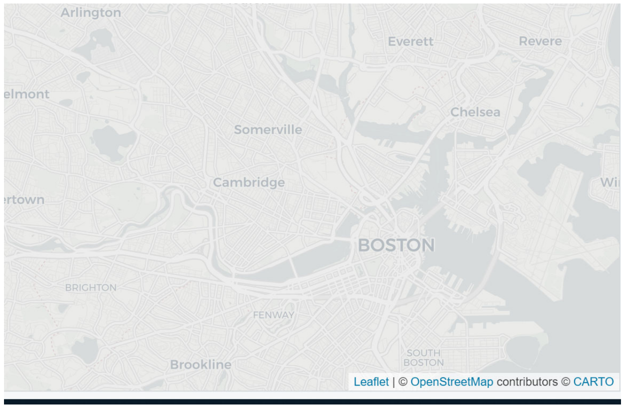
```
#making leaflet map for February 2020
leafmap.feb<- leaflet(locfeb2020) %>%
  addProviderTiles(providers$CartoDB.Positron,
                   options = providerTileOptions(opacity = 0.5)) %>%
```

WHAT'S HAPPENING IN THE CODE?

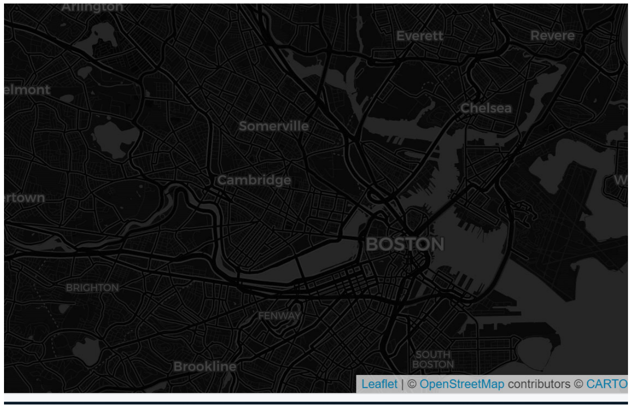
1. Calling 'leaflet' function and inputting data
2. Defining basemap



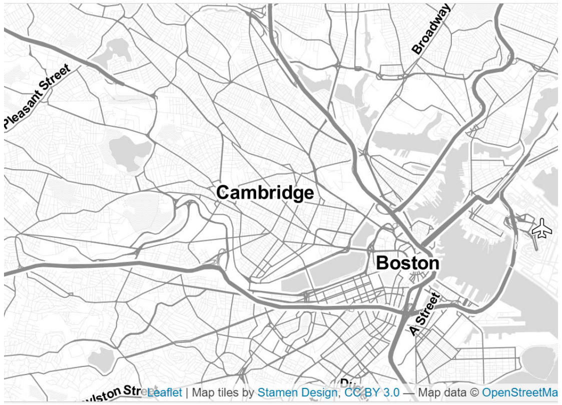
```
#making leaflet map for February 2020
leafmap.feb<- leaflet(locfeb2020) %>%
  addProviderTiles(providers$CartoDB.Positron,
                   options = providerTileOptions(opacity = 0.5)) %>%
```



providers$CartoDB.Positron



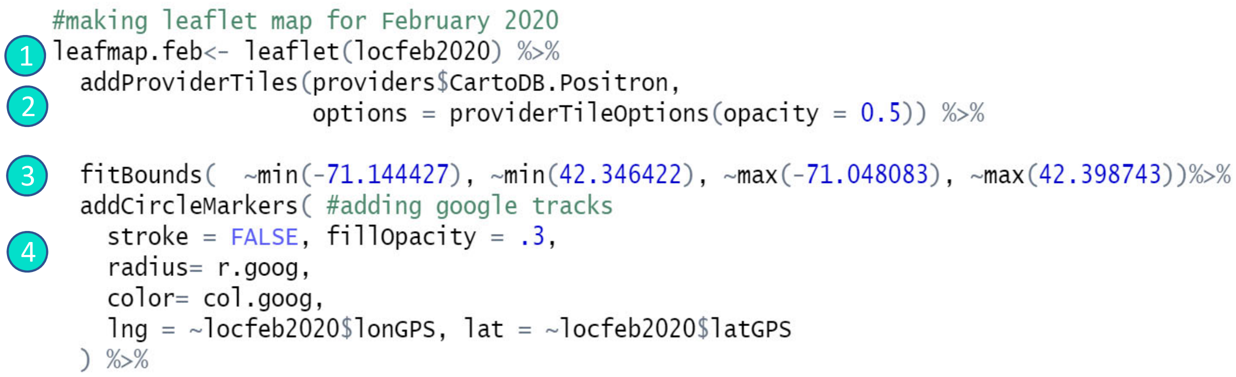
providers$CartoDB.DarkMatter


Pleasant Street
Broadway
Cambridge
Boston
A Street
lston Str
Leaflet | Map tiles by Stamen Design, CC BY 3.0 — Map data © OpenStreetMa

providers$Stamen.TonerLite

WHAT’S HAPPENING IN THE CODE

1. Calling ‘leaflet’ function and inputting data
2. Defining basemap
3. Setting extent of where in the world the resulting map will ‘zoom’ as default
4. Add symbols which are circular markers
    - Define their stroke, opacity, radius, color and what longitude and latitude each circle will be displayed at



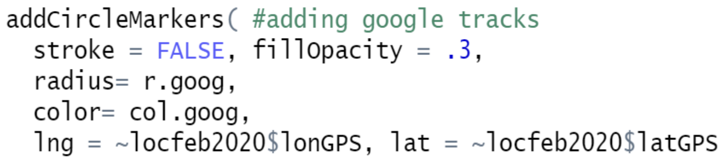
```r
#making leaflet map for February 2020
leafmap.feb<- leaflet(locfeb2020) %>%
  addProviderTiles(providers$CartoDB.Positron,
                   options = providerTileOptions(opacity = 0.5)) %>%

  fitBounds(  ~min(-71.144427), ~min(42.346422), ~max(-71.048083), ~max(42.398743))%>%
  addCircleMarkers( #adding google tracks
    stroke = FALSE, fillOpacity = .3,
    radius= r.goog,
    color= col.goog,
    lng = ~locfeb2020$lonGPS, lat = ~locfeb2020$latGPS
  ) %>%
```

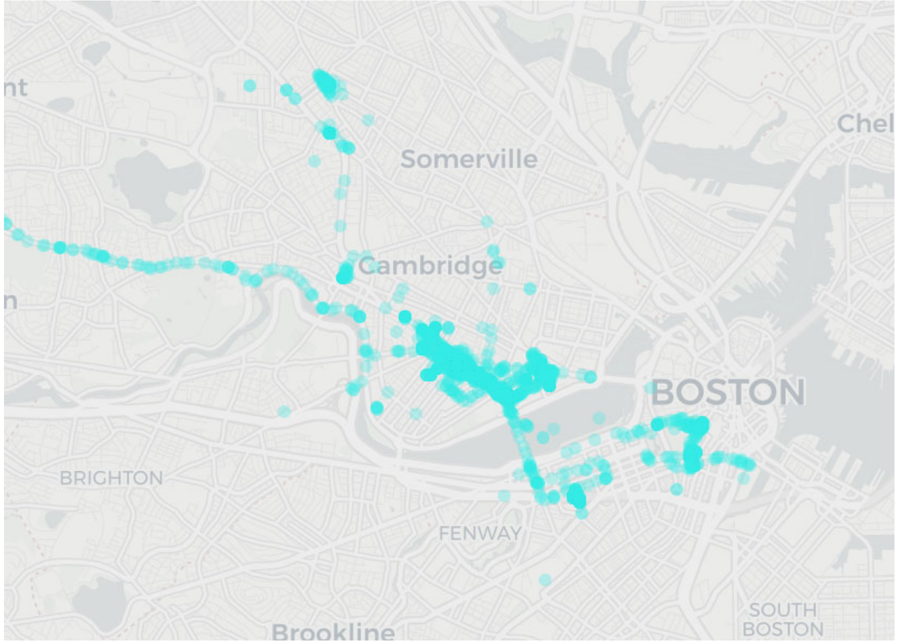
WHAT’S HAPPENING IN THE CODE

4. Add symbols which are circular markers
    - Define their stroke, opacity, radius, color and what longitude and latitude each circle will be displayed at

4



```
addCircleMarkers( #adding google tracks
  stroke = FALSE, fillOpacity = .3,
  radius= r.goog,
  color= col.goog,
  lng = ~locfeb2020$lonGPS, lat = ~locfeb2020$latGPS
```

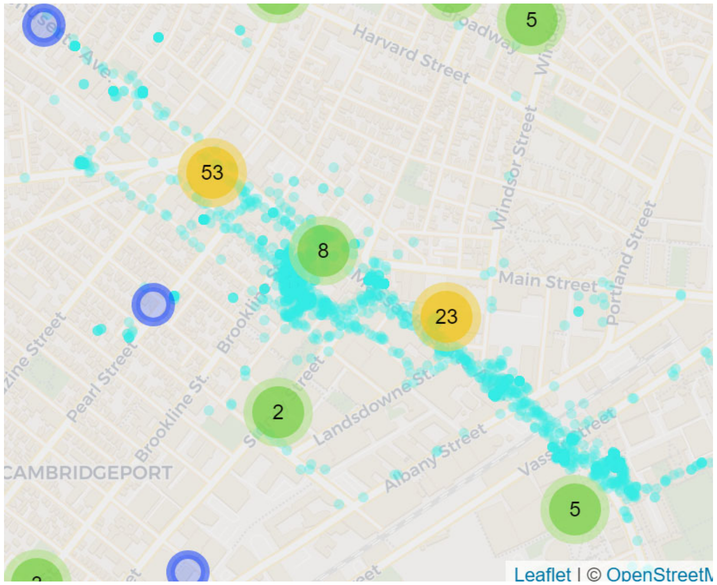


WHAT'S HAPPENING IN THE CODE

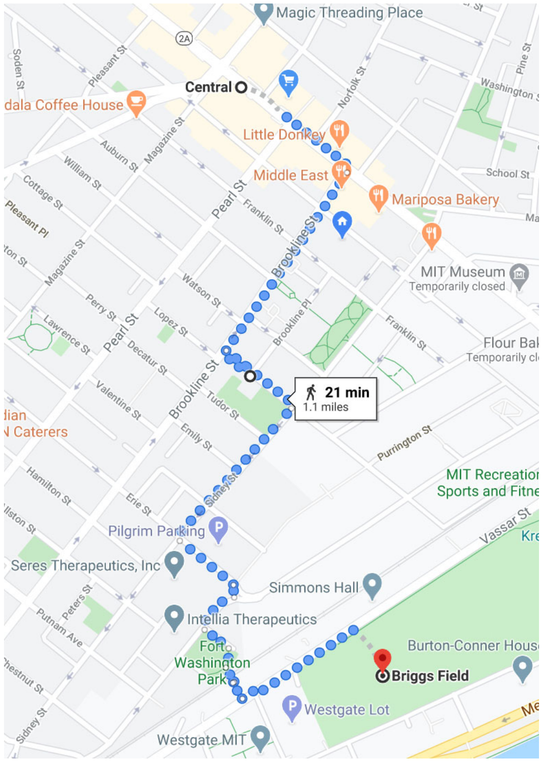
We can add different layers to the map by piping %>%

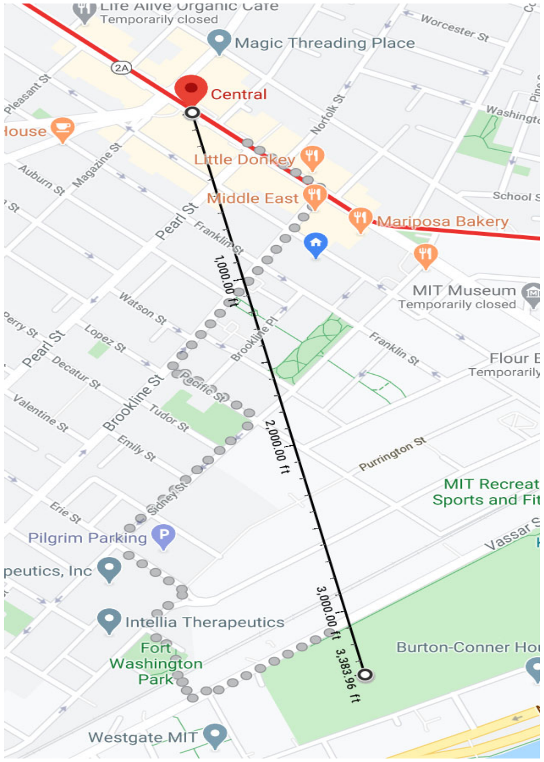
```
addCircleMarkers( #adding google tracks
  stroke = FALSE, fillOpacity = .3,
  radius= radius,
  color= col.goog,
  lng = ~locfeb2020$lonGPS, lat = ~locfeb2020$latGPS
) %>%
addCircleMarkers(   #adding yelp data

  lng = ~yelp$longitude, lat = ~yelp$latitude,
  clusterOptions = markerClusterOptions()
)
```



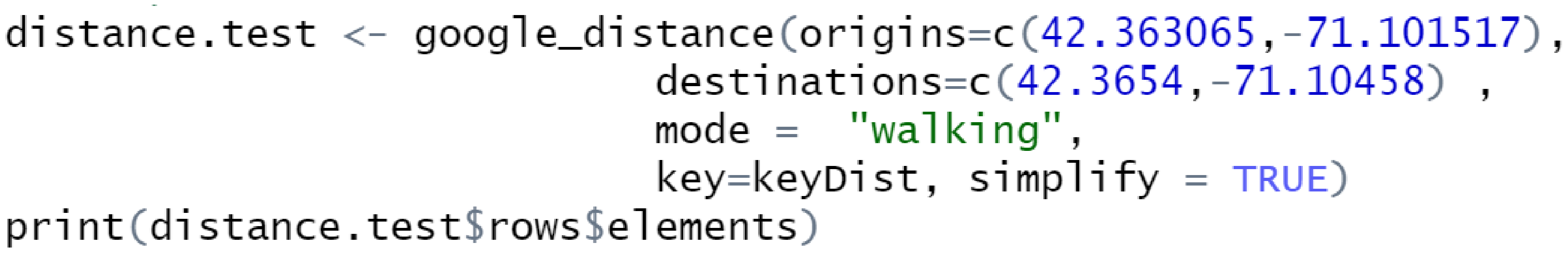
NETWORK vs EUCLIDEAN





GOOGLEWAY LIBRARY IN R

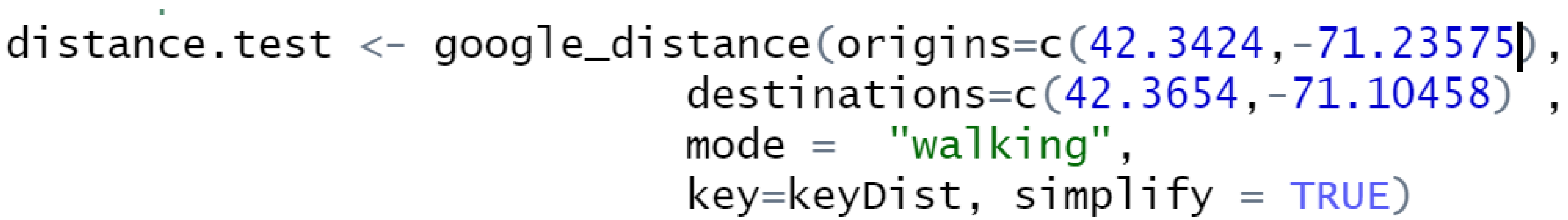
- Use same algorithm Google uses to calculate distance
- Access through R and Google API Key



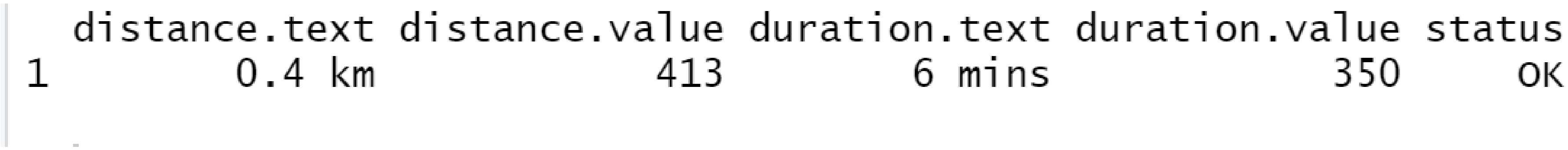
```r
distance.test <- google_distance(origins=c(42.363065,-71.101517),
                           destinations=c(42.3654,-71.10458) ,
                           mode =  "walking",
                           key=keyDist, simplify = TRUE)
print(distance.test$rows$elements)
```

WHAT'S HAPPENING IN THE CODE?

- Pick a lat/long as origin
- Pick a lat/long as destination
- Calculate distance via desired 'mode'
- Set API Key value



```
distance.test <- google_distance(origins=c(42.3424,-71.23575),
                                 destinations=c(42.3654,-71.10458) ,
                                 mode =  "walking",
                                 key=keyDist, simplify = TRUE)
```



```
  distance.text distance.value duration.text duration.value status
1        0.4 km            413        6 mins            350     OK
```

Make a loop that repeats this process as many times as there are rows in the Yelp dataset

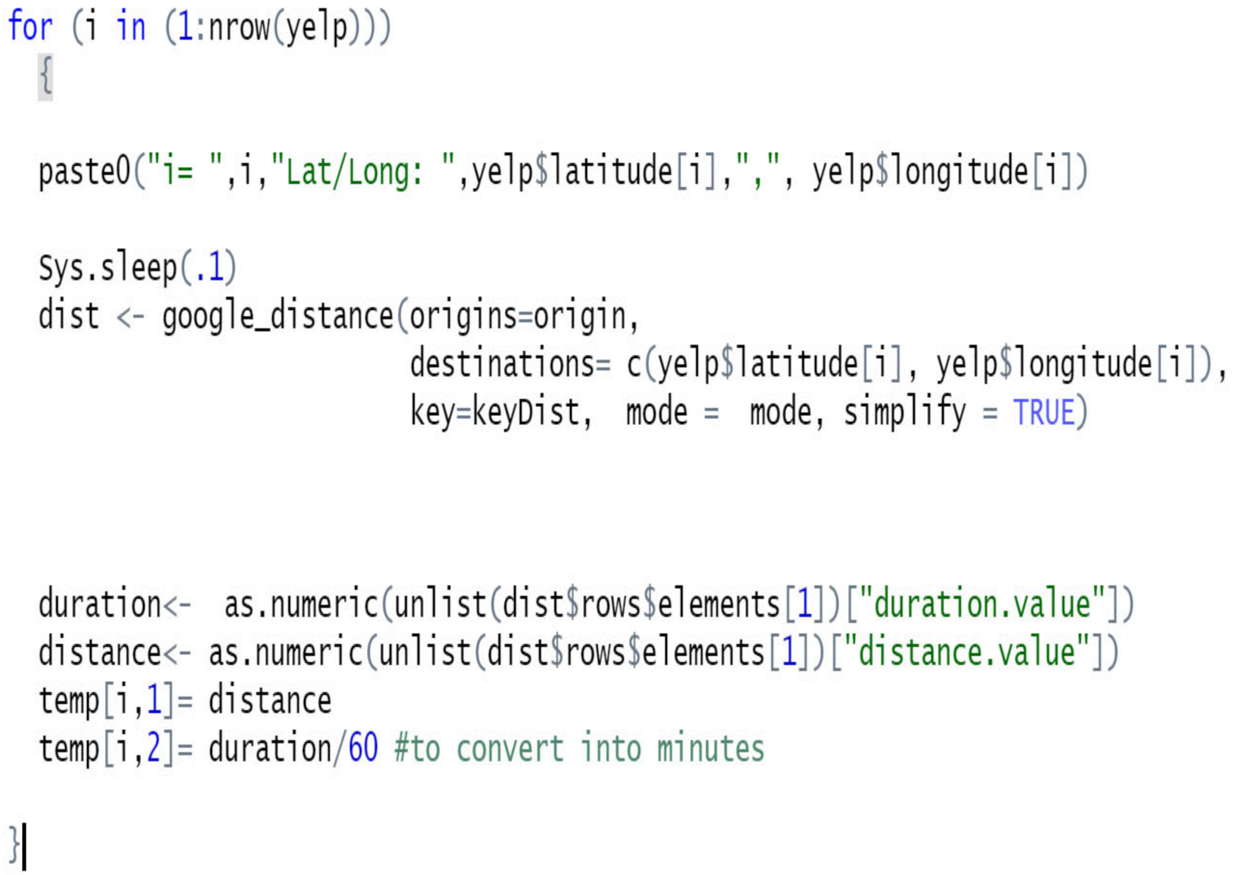
Each i- represents a row index e.g. when i- is 1, the loop is on the first row of the yelp dataset

Set an origin point

Set a destination (i-th row of yelp dataset)

Create a variable that extracts the duration/distance from results

Appends distance into row i of column 1 and duration into row i of column 2 duration



```
for (i in (1:nrow(yelp)))
  {

  paste0("i= ",i,"Lat/Long: ",yelp$latitude[i],",", yelp$longitude[i])

  Sys.sleep(.1)
  dist <- google_distance(origins=origin,
                          destinations= c(yelp$latitude[i], yelp$longitude[i]),
                          key=keyDist,  mode =  mode, simplify = TRUE)



  duration<-  as.numeric(unlist(dist$rows$elements[1])["duration.value"])
  distance<- as.numeric(unlist(dist$rows$elements[1])["distance.value"])
  temp[i,1]= distance
  temp[i,2]= duration/60 #to convert into minutes

}
```

DISADVANTAGES OF DIGITAL TRACE DATA

No control over what and is not available or understanding of how it is stored

- e.g. is an establishment showing up under the search term of ‘food’ vs ‘restaurants’

Validity of the inferences

- can only observe behavior, not understand intentionality
- behavior being observed on social media is not ‘natural’ or non-reactive’

Conflict with current standards of informed consent and privacy