GIS Lab Assignment:

POI Buffer Analysis Using the CeTLeR API

Objective:

This lab involves using the CeTLeR API to download and analyse Points of Interest (POIs) related to bus stops and restaurants within a specified area in Germany or Sweden. The goal is to apply spatial buffering techniques to examine the proximity of restaurants (target features) to bus stops (service features) and visualize the results on a map.

Materials Needed:

- Access to the CeTLeR API (https://apis.cetler.se)
- Use R in handling API data and spatial analysis.
- Internet connection

Libraries:

- library(httr)
- library(jsonlite)
- library(sf)
- library(tmap)
- library(leaflet)

Procedure:

1. API Data Retrieval:

- Choose an area of interest either in Germany or Sweden.
- Use the CeTLeR API to download data for two types of POIs: bus stops and restaurants.

2. Buffering Technique Application:

- Apply a buffering technique available via the CeTLeR API. If you select a circular buffer, ensure that the radius is no less than 1500 meters.

3. Buffer Analysis for Spatial Relationships:

- Designate restaurants as the target features and bus stops as the service features.
- Create a 400-meter radius buffer around each restaurant.
- Analyse each buffer to determine if it contains at least one bus stop within its confines.

4. Buffer Visualization:

Visualize the buffer zones on your map. Use the following color coding:

- Green Buffer: Indicates the presence of at least one bus stop within the 400-meter radius of a restaurant.
- Red Buffer: Indicates no bus stop presence within the 400-meter radius.
- Ensure each buffer is clearly represented in either green or red based on the presence of service features.

5. Map Presentation:

- Project all the results onto a map.
- Include a legend that clearly identifies the target features (restaurant) and service features (bust stops).
- Ensure the map is legible and appropriately scaled to showcase all relevant data points and buffers.

6. Analysis and Conclusion:

- Examine the distribution of restaurants and their proximity to bus stops.
- Discuss patterns observed and potential implications for business or urban planning.

Submission Requirements:

- Submit a digital copy of the map with the applied buffers and legend.
- Include a brief report (1 page) summarizing your methodology, findings, and any challenges encountered during the lab.

Tips:

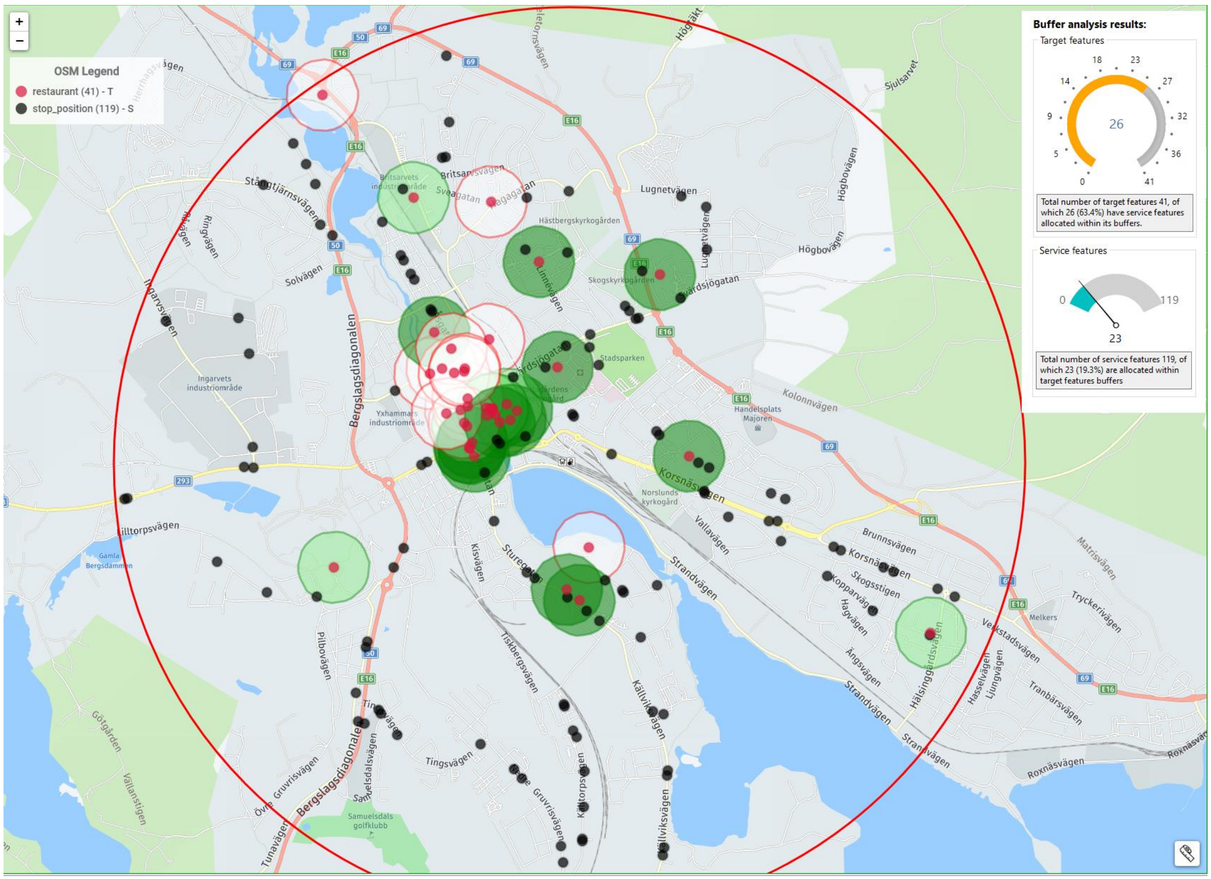
- To download Restaurant, the key and value are:
  - Key: amenity
  - Value: restaurant
- To download bust stops, the key and value are:
  - Key: public_transport
  - Value: stop_position
- Ensure to utilize the code sample provided on the API page at http://apis.cetler.se.
- Additionally, establish a color palette using the following code:

```
colors <- colorRampPalette(brewer.pal(9, "YlOrRd"))(length(unique(data$value)))
```

- To determine if there's a bus stop within the restaurant buffer, calculate the distance between the buffer's centre and the bus stop employing the Haversine formula.
- Make two calls to the API: one to retrieve the points of interest (POI) for restaurants and the other to fetch bus stop data.

Output:

Example of the assignment output.



Buffer analysis results:

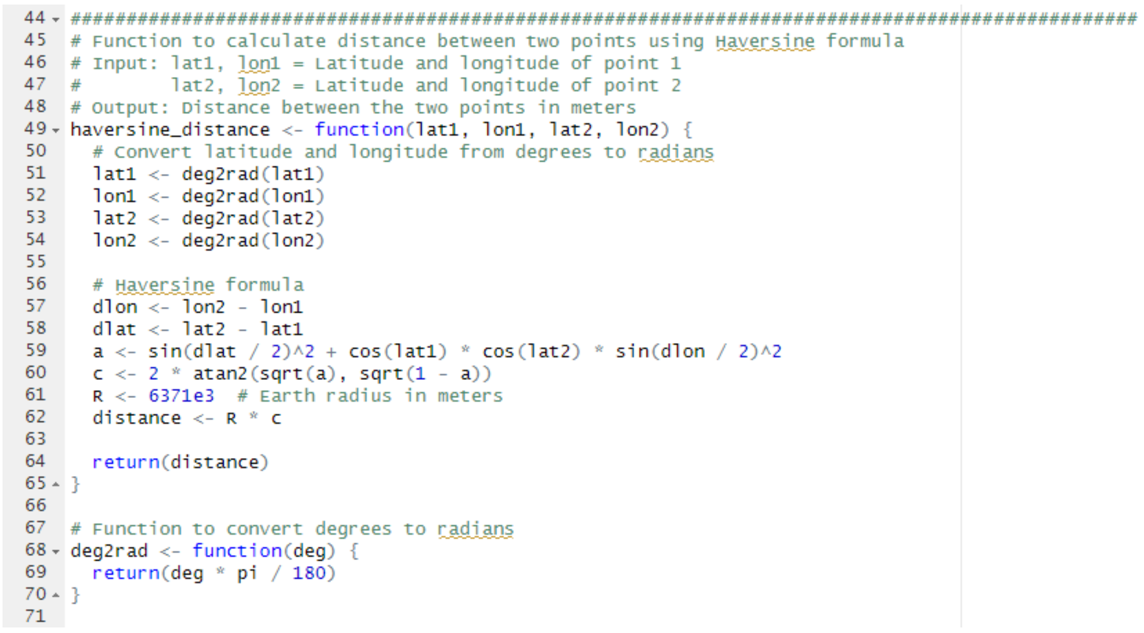
Total number of target features 41, of which 26 (63.4%) have service features allocated within its buffers.

Total number of service features 119, of which 23 (19.3%) are allocated within target features buffers

Required methods:

Haversine formula:

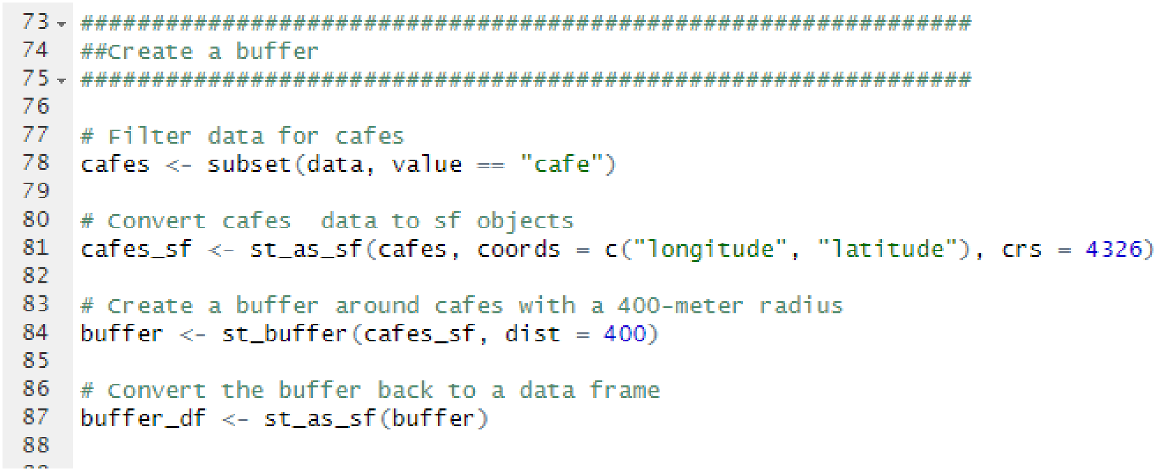
To find the distance between two points.



```r
##############################################################################################
# Function to calculate distance between two points using Haversine formula
# Input: lat1, lon1 = Latitude and longitude of point 1
#        lat2, lon2 = Latitude and longitude of point 2
# Output: Distance between the two points in meters
haversine_distance <- function(lat1, lon1, lat2, lon2) {
  # Convert latitude and longitude from degrees to radians
  lat1 <- deg2rad(lat1)
  lon1 <- deg2rad(lon1)
  lat2 <- deg2rad(lat2)
  lon2 <- deg2rad(lon2)

  # Haversine formula
  dlon <- lon2 - lon1
  dlat <- lat2 - lat1
  a <- sin(dlat / 2)^2 + cos(lat1) * cos(lat2) * sin(dlon / 2)^2
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  R <- 6371e3  # Earth radius in meters
  distance <- R * c

  return(distance)
}

# Function to convert degrees to radians
deg2rad <- function(deg) {
  return(deg * pi / 180)
}
```

Create the buffers:

The following code filters the data obtained from the API and proceeds to create a buffer around each café with a radius of 400 units.



```r
###############################################################
##Create a buffer
###############################################################

# Filter data for cafes
cafes <- subset(data, value == "cafe")

# Convert cafes  data to sf objects
cafes_sf <- st_as_sf(cafes, coords = c("longitude", "latitude"), crs = 4326)

# Create a buffer around cafes with a 400-meter radius
buffer <- st_buffer(cafes_sf, dist = 400)

# Convert the buffer back to a data frame
buffer_df <- st_as_sf(buffer)
```