Berlin: Extracting Data of Cultural and Religious Locations from OSM

Orla Mallon and Nicole Dwenger

11/05/2021

This script retrieves data of cultural locations and religious locations in Berlin from OSM.

1. Preparation

1.1. Packages

```
#Loading up the packages
library(tidyverse)
library(dplyr)
library(osmdata)
library(sf)
library(leaflet)
library(here)
```

1.2. Setting up the bounding box for Berlin

```
#Create a bounding box
bb <- getbb('berlin de', format_out = 'polygon')[1]
```

1.3. Loading the Berlin Borough geometries

```
berlin_raw <- st_read(here("berlin", "raw_data", "bezirke", "bezirksgrenzen.shp"))
```

```
## Reading layer 'bezirksgrenzen' from data source '/Users/nicoledwenger/Documents/University/6SEMESTER/
## Simple feature collection with 12 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 13.08835 ymin: 52.33825 xmax: 13.76116 ymax: 52.67551
## Geodetic CRS:  WGS 84
```

```r
# Selecting relevant variables and transforming geometry of polygons into coordinate system for leaflet
# This is the same as the OSM CRS
berlin_geo = berlin_raw %>%
  dplyr::select("name" = Gemeinde_n,
                "geometry" = geometry) %>%
  st_transform(25833)

# Saving the crs to use for further processing
crs_de = st_crs(berlin_geo)
crs_de
```

```
## Coordinate Reference System:
##   User input: EPSG:25833
##   wkt:
## PROJCRS["ETRS89 / UTM zone 33N",
##     BASEGEOGCRS["ETRS89",
##         DATUM["European Terrestrial Reference System 1989",
##             ELLIPSOID["GRS 1980",6378137,298.257222101,
##                 LENGTHUNIT["metre",1]]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]],
##         ID["EPSG",4258]],
##     CONVERSION["UTM zone 33N",
##         METHOD["Transverse Mercator",
##             ID["EPSG",9807]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",15,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.9996,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",500000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["unknown"],
##         AREA["Europe - 12°E to 18°E and ETRS89 by country"],
##         BBOX[46.4,12,84.01,18.01]],
##     ID["EPSG",25833]]
```

2. Culture: Museums, Theatres & Nightlife

2.1. Defining function to get both the points and polygons exracted from OSM

OSM returns some locations as points and some as polygons, where the polygons simply define the shape of a building. We would like all locations to be points. Thus, we get all the points as they are, and we get the polygons, but transform them into the British CRS, get the centroids of each of the locations and then transform it back to 4326 for Leaflet. However, we only keep the locations which do not have the no name or the name name.

```r
get_culture_locations = function(raw_osm_data, crs, type, type_name){

  # From the raw osm data get the points
  points = raw_osm_data$osm_points %>%
    # Keep only the name and geometry column
    dplyr::select("name", "geometry") %>%
    # Drop the names wth NA's
    drop_na(name) %>%
    # Keep only the unique names
    distinct(., name, .keep_all=T)

  #  From the raw osm data get the polygons
  polygons = raw_osm_data$osm_polygons %>%
    # Keep only the name and geometry column
    dplyr::select("name", "geometry") %>%
    # Drop the names wth NA's
    drop_na(name) %>%
    # Transform into projected crs
    st_transform(crs) %>%
    # Replace polygons with centroids
    mutate(geometry = st_centroid(geometry)) %>%
    # Transform back to 4326 for leaflet
    st_transform(crs=4326) %>%
    # Keep only the unique names
    distinct(., name, .keep_all=T)

  # Put the points and polygons together
  locations = rbind(points, polygons) %>%
    # And keep only the unique ones
    distinct(., name, .keep_all=T)

  # Reset the index
  rownames(locations) = NULL
  # Add column with type
  locations$type <- as.character(type)
  # Add the type to the name
  locations$name = paste0(type_name, ": ", locations$name)

  # Return the locations
  locations
}
```

2.2. Museums

An institution which normally has exhibitions on scientific, historical, cultural topics, etc. Typically open to the public as a tourist attraction, museums may be more heavily involved in acquisitions, conservation or research.

```
# Get all output for museums for the bb box of Berlin
raw_museums <- opq(bbox = bb) %>%
    add_osm_feature(key = 'tourism', value = 'museum') %>%
    osmdata_sf() %>% # Make into sf object
    trim_osmdata(bb) # Trim by bounding box

# Get the points and polygons (transformed to points by getting the centroids)
museums = get_culture_locations(raw_museums, crs_de, "museum", "Museum")
# Check crs
st_crs(museums)$input
```

```
## [1] "EPSG:4326"
```

2.3. Theatres

A place where live performances occur, such as plays, musicals and formal concerts.

```
# Get all output for theatres for the bb box of Berlin
raw_theatres <- opq(bbox = bb) %>%
    add_osm_feature(key = 'amenity', value = 'theatre') %>%
    osmdata_sf() %>%  # Make into sf object
    trim_osmdata(bb) # Trim by bounding box

# Get the points and polygons (transformed to points by getting the centroids)
theatres = get_culture_locations(raw_theatres, crs_de, "theatre", "Theatre")
# Check crs
st_crs(theatres)$input
```

```
## [1] "EPSG:4326"
```

2.4. Nightclubs

A place to dance and drink at night. Also known as a disco.

```
# Get all output for nightclubs for the bb box of Berlin
raw_nightclubs <- opq(bbox = bb) %>%
    add_osm_feature(key = 'amenity', value = 'nightclub') %>%
    osmdata_sf() %>% # Make into sf object
    trim_osmdata(bb) # Trim by bounding box

## Get the points and polygons (transformed to points by getting the centroids)
nightclubs = get_culture_locations(raw_nightclubs, crs_de, "nightlife", "Nightclub")
# Check crs
st_crs(nightclubs)$input
```

```
## [1] "EPSG:4326"
```

2.5. Bars

Also a late night drinking place. Perhaps less emphasis on dancing/disco. Perhaps less likely to require entry fee

```r
# Get all output for bars for the bb box of Berlin
raw_bars <- opq(bbox = bb) %>%
  add_osm_feature(key = 'amenity', value = 'bar') %>%
  osmdata_sf()%>%   # Make into an sf object
  trim_osmdata(bb)  # Trim to the Berlin city bb

## Get the points and polygons (transformed to points by getting the centroids)
bars = get_culture_locations(raw_bars, crs_de, "nightlife", "Bar")
# Check crs
st_crs(bars)$input
```

```
## [1] "EPSG:4326"
```

2.6. Pubs

A more traditional style drinking place. Much less emphasis on dancing/disco. More emphasis on beer.

```r
# Get all output for pubs for the bb box of Berlin
raw_pubs <- opq(bbox = bb) %>%
  add_osm_feature(key = 'amenity', value = 'pub') %>%
  osmdata_sf() %>%  # Make into an sf object
  trim_osmdata(bb)  # Trim to the Berlin city bb

## Get the points and polygons (transformed to points by getting the centroids)
pubs = get_culture_locations(raw_pubs, crs_de, "nightlife", "Pub")
# Check crs
st_crs(pubs)$input
```

```
## [1] "EPSG:4326"
```

2.7. Join all locations together and save

```r
# Rbind all together
culture_locations = rbind(museums, theatres, nightclubs, bars, pubs)
# Save as csv and RDS
#write.csv(culture_locations, "berlin/preprocessed_data/berlin_culture_locations.csv")
#saveRDS(culture_locations, "berlin/preprocessed_data/berlin_culture_locations.rds")
```

2.8. Calculating the number of locations of each type per borough

```r
# Turn the centroids back into the british CRS
culture_locations_proj = st_transform(culture_locations, crs=crs_de)
```

```
# Look through each of the types, and for each get the count for each polygon and add them to the data
culture_counts = berlin_geo
for (unqiue_type in unique(culture_locations_proj$type)){
  # Subset only the locations for the unqiue_type (museum, theatre, nighlife)
  type_subset = filter(culture_locations_proj, type == unqiue_type)
  # Count how many locations lie in each borough
  counts = lengths(st_intersects(culture_counts, type_subset))
  # Create column name
  column_name = paste0("n_", unqiue_type)
  # Cdd counts to the column with column name
  culture_counts[column_name] = counts
}

# Drop the geometry column
culture_counts_csv <- st_drop_geometry(culture_counts)

# Save as csv file and RDS
#write.csv(culture_counts_csv, "berlin/preprocessed_data/berlin_culture_counts.csv", row.names = F)
#saveRDS(culture_counts_csv, "berlin/preprocessed_data/berlin_culture_counts.rds")
```

3. Religion: Places of Worship

3.1. Defining function to get both the points and polygons exracted from OSM

OSM returns some locations as points and some as polygons, where the polygons simply define the shape of a building. We would like all locations to be points. Thus, we get all the points as they are, and we get the polygons, but transform them into the British CRS, get the centroids of each of the locations and then transform it back to 4326 for Leaflet. However, we only keep the locations which do not have the no name or the name name. This function is similar to the one for cultural locations, just a bit more complex to avoid problems with religions, where no place of worship was found.

```
get_religion_locations = function(raw_osm_data, crs, religion){

  # Get the raw points and polygons
  raw_points = raw_osm_data$osm_points
  raw_polygons = raw_osm_data$osm_polygons

  # If there are points, and the data has the column name "name" and there is at least one location whe
  if (!is.null(raw_points) & "name" %in% colnames(raw_points) & nrow(raw_points[!is.na(raw_points$name)
     points = raw_points %>%
      # Select the relevant columns
      dplyr::select("name", "geometry") %>%
      # Drop rows where the name is NA
      drop_na(name) %>%
      # Keep only unique names
      distinct(., name, .keep_all=T)
  } else {
    # Otherwise, return NULL
    points = NULL
  }

  # If there are polygons and the data has the column name "name" and there is at least one location wh
```

```r
  if (!is.null(raw_polygons) & "name" %in% colnames(raw_polygons) & nrow(raw_polygons[!is.na(raw_polygo
  polygons = raw_polygons %>%
    # Select only relevant columns
    dplyr::select("name", "geometry") %>%
    # Drops rows where the name is NA
    drop_na(name) %>%
    # Transform into the given crs
    st_transform(crs) %>%
    # Replace geometry of polygon with centroid
    mutate(geometry = st_centroid(geometry)) %>%
    # Transform back to 4326 for leaflet
    st_transform(crs=4326) %>%
    # Keep only the rows with unique names
    distinct(., name, .keep_all=T)
  } else {
    # Otherwise, return NULL
    polygons = NULL
  }

  # If no points were found, the locations are only the ones extracted from the polygons
  if (is.null(points)){
    locations = polygons
  # If no polygons were found, the locations are only the ones extracted from the points
  } else if (is.null(polygons)){
    locations = points
  # If both were found, bind them together and only keep the unique names
  } else if (!is.null(points) & !is.null(polygons)){
    locations = rbind(points, polygons) %>% distinct(., name, .keep_all=T)
  }

  # If the locations are not null, return them
  if (!is.null(locations)){
    # Reset index
    rownames(locations) = NULL
    # Add column with religion type
    locations$religion = as.character(religion)
    # Return locations
    locations
  }
}
```

3.2. Buddhist

```r
# Get the raw data for all buddhist places of worship
raw_buddhist <- opq(bbox = bb) %>%
  add_osm_feature(key = "amenity", value = "place_of_worship") %>%
  add_osm_feature(key = 'religion', value = 'buddhist') %>%
  osmdata_sf() %>%
  trim_osmdata(bb)

# Get the points and polygons (transformed to points by getting the centroids)
buddhist = get_religion_locations(raw_buddhist, crs_de, "buddhist")
```

```r
# Check crs
st_crs(buddhist)$input
```

```
## [1] "EPSG:4326"
```

3.3. Hindu

```r
# Get the raw data for all hindu places of worship
raw_hindu <- opq(bbox = bb) %>%
  add_osm_feature(key = "amenity", value = "place_of_worship") %>%
  add_osm_feature(key = 'religion', value = 'hindu') %>%
  osmdata_sf() %>%
  trim_osmdata(bb)

# Get the points and polygons (transformed to points by getting the centroids)
hindu = get_religion_locations(raw_hindu, crs_de, "hindu")
# Check crs
st_crs(hindu)$input
```

```
## [1] "EPSG:4326"
```

3.4. Jewish

```r
# Get the raw data for all jewish places of worship
raw_jewish <- opq(bbox = bb) %>%
  add_osm_feature(key = "amenity", value = "place_of_worship") %>%
  add_osm_feature(key = 'religion', value = 'jewish') %>%
  osmdata_sf() %>%
  trim_osmdata(bb)

# Get the points and polygons (transformed to points by getting the centroids)
jewish = get_religion_locations(raw_jewish, crs_de, "jewish")
# Check crs
st_crs(jewish)$input
```

```
## [1] "EPSG:4326"
```

3.5. Muslim

```r
# Get the raw data for all muslim places of worship
raw_muslim = opq(bbox = bb) %>%
  add_osm_feature(key = "amenity", value = "place_of_worship") %>%
  add_osm_feature(key = 'religion', value = 'muslim') %>%
  osmdata_sf() %>%
  trim_osmdata(bb)

# Get the points and polygons (transformed to points by getting the centroids)
```

```
muslim = get_religion_locations(raw_muslim, crs_de, "muslim")
# Check crs
st_crs(muslim)$input
```

```
## [1] "EPSG:4326"
```

3.6. Sikh

```
# Get the raw data for all sikh places of worship
raw_sikh = opq(bbox = bb) %>%
  add_osm_feature(key = "amenity", value = "place_of_worship") %>%
  add_osm_feature(key = 'religion', value = 'sikh') %>%
  osmdata_sf() %>%
  trim_osmdata(bb)

# Get the points and polygons (transformed to points by getting the centroids)
sikh = get_religion_locations(raw_sikh, crs_de, "sikh")
# Check crs
st_crs(sikh)$input
```

```
## [1] NA
```

```
# NONE FOUND
```

3.7. Protestant

Getting all the largest protestant denominations from here: https://wiki.openstreetmap.org/wiki/Key:denomination

```
# Define all the protestant denominations
protestant_denominations = c("protestant", "adventist", "anglican", "baptist", "churches_of_christ", "d
                             "episcopal", "evangelical", "evangelical_covenant", "exclusive_brethren",
                             "methodist", "moravian", "mormon",  "pentecostal","pentecostal", "presbyte
                             "strict_baptist", "uniting")

# Create empty data frame to fill with loop
protestant = data.frame()

# Loop through all denominations and get the data
for (denomination in protestant_denominations){

  # Get the raw locations for the denomination
  raw_denomination_loc = opq(bbox = bb) %>%
    add_osm_feature(key = "amenity", value = "place_of_worship") %>%
    add_osm_feature(key = 'religion', value = 'christian') %>%
    add_osm_feature(key = 'denomination', value = denomination) %>%
    osmdata_sf() %>%
    trim_osmdata(bb)

  # Process the raw locations of the denomination
```

```r
  denomination_loc = get_religion_locations(raw_denomination_loc, crs_de, "protestant")

  if (!is.null(denomination_loc)){
    protestant = rbind(protestant, denomination_loc)}
}

# Keep only the unique ones after getting them from all denominations
protestant = protestant %>% distinct(., name, .keep_all=T)
```

3.8. Chatholic

```r
# Define list of catholic denominations of interest
catholic_denominations = c("roman_catholic", "catholic", "orthodox")

# Create empty data frame to store catholic locations
catholic = data.frame()

# Loop through the denominations to get all catholic locations
for (denomination in catholic_denominations){

  # Get the raw osm data for the denomination
  raw_denomination_loc = opq(bbox = bb) %>%
    add_osm_feature(key = "amenity", value = "place_of_worship") %>%
    add_osm_feature(key = 'religion', value = 'christian') %>%
    add_osm_feature(key = 'denomination', value = denomination) %>%
    osmdata_sf() %>%
    trim_osmdata(bb)

  # Get the point and polygon locations
  denomination_loc = get_religion_locations(raw_denomination_loc, crs_de, "catholic")

  # If it is not null, bind it to the data frame
  if (!is.null(denomination_loc)){
    catholic = rbind(catholic, denomination_loc)
  }
}

# Keep only the unqiue ones
catholic = catholic %>% distinct(., name, .keep_all=T)
```

3.9. Bind all religions together

```r
religion_locations = rbind(buddhist, hindu, jewish, muslim, protestant, catholic)

# Save as csv and RDS
#write.csv(religion_locations, "berlin/preprocessed_data/berlin_religion_locations.csv")
#saveRDS(religion_locations, "berlin/preprocessed_data/berlin_religion_locations.rds")
```

3.10. Calculate the religious locations per borough

```r
# Turn the centroids back into the british CRS
religion_locations_proj = st_transform(religion_locations, crs=crs_de)

# Look through each of the types, and for each get the count for each polygon and add them to the data
religion_counts = berlin_geo
for (unqiue_religion in unique(religion_locations_proj$religion)){
  religion_subset = filter(religion_locations_proj, religion == unqiue_religion)
  counts = lengths(st_intersects(religion_counts, religion_subset))
  column_name = paste0("n_", unqiue_religion)
  religion_counts[column_name] = counts
}

# Drop the geometry column
religion_counts_csv <- st_drop_geometry(religion_counts)

# Save as csv file and RDS
#write.csv(religion_counts_csv, "berlin/preprocessed_data/berlin_religion_counts.csv", row.names = F)
#saveRDS(religion_counts_csv, "berlin/preprocessed_data/berlin_religion_counts.rds")
```