

Project document

1. Data description:

In general, the data consists of different users trajectories, based on GPS logs, submitted by each user. The data origins from two sources – one is the OpenStreetMap project, from which were downloaded the trajectories belonging to Europe, and the other is the GeoLife project of the Asian Microsoft group, which contains trajectories from Asia and U.S. Both of them, are open source data.

The available data regarding a trajectory, is to which user it belongs, and the sequence of geographical coordinates it contains with a time stamp for each of them.

This data serves me in my thesis - its first goal is the discovery of trajectory patterns from the trajectory set of each user. The findings of the above will be used to complete the main goal which is the prediction of a next stay location on the user's trajectory.

In this visualization project, the data which is used is the collection of the start and end points of each of the trajectories of some user. The idea is to be able to see the grouping of different types of locations the user is coming from, and going to, as will be detailed below.

The demonstration is shown on a user with 512 data (start and end) points and 463 semantic locations (as extracted using R "osmar" library – as will be further explained), in total there are 975 points to visualize for this specific user. There are users with much more data points, and users with less.

2. User tasks and purpose:

For this particular visualization, the user has a table of geographic coordinates – the latitude and longitude - of each start and end point, extracted from the user's trajectories.

The user has several purposes:

- a. To be able to see on a map the basics:
 1. To what country/city/street this coordinates belong – there will be trajectories of people which are traveling only in one city, while others may have trajectories from different countries.
 2. How the data is spread on the map – are the points close to each other or spread.
- b. To be able to see, what are the semantic locations near by this data points.

We need to search for a near location for two reasons: first is because GPS devices have a few meters errors, and second is that it is reasonable to think that the user did not turn on or off his recording GPS device exactly in the moment he got out of some place or arrived to some other place, therefore this start or end points will not “hit” exactly at a concrete semantic location (e.g. shop, bank, school etc.) but somewhere near the true location. How near is near enough is the user’s decision to make. The user can assign different distance thresholds and see how it affects the results, by changing in the code the value of the variable “search_dist” (was intended to be an input parameter in Shiny, but the final result doesn’t include Shiny).

- c. Evaluating the output of a clustering algorithm - To be able to see how the different data points group on a map based on a clustering algorithm which divides them to groups (clusters) and how the clustering results change as a function of the algorithms parameters. In this project was implemented a simple but well known clustering algorithm - the K-Means. This algorithm has only one parameter the user needs to set, and that is the value of K, meaning, the user needs to choose to how many clusters the data will be divided. The user needs to set this value to the variable “num_of_clusters” which appears in the beginning of the code. The value of K needs to be in the range of 2-10. There is no meaning to setting the number of clusters to 1, and the max value should not be higher than 10 just as a personal choice.
- d. Summarizing the data regarding the semantic locations found within the geographical area, defined by each cluster.

3. Visual Design Iterations:

➤ First iteration: (It’s only the beginning - only what was implemented in Shiny)

Initially the project was meant to be implemented via the Shiny library of R, in order for it to be more interactive and more convenient to use.

Next will be presented the screen shots of the first part that did work via Shiny application.

a. Visual mapping of data-to-visual attributes:

- Geographical points (longitude, latitude) → location of circle markers on the map
- Clustering results (to which cluster each data point belongs) → color of the marker

The color palette is a color brewer palette for categorical data which is called "Set3, with 10 colors because 10 is the maximum number of clusters I choose.

b. Code

```
library(shiny)
library(RColorBrewer)
library(clusterSim)
library(leaflet)
library(osmar)
library(sp)
library(maptools)
library(rgdal)
library(rgeos)
library(htmltools)
library(dplyr)
library(magrittr)

data <- read.csv("")
selectedData <- data.frame(data$Lat, data$Lon)
pal <- colorFactor(palette= brewer.pal(10, "Set3"), domain = c(1:10))

ui <- fluidPage(
  headerPanel('Locations clustering by k-means'),
  sidebarPanel(
    sliderInput('clusters', 'Number of clusters', 2, min = 2, max = 10),
    sliderInput('dist', 'Search Distance', 100, min = 10, max = 200)),
  mainPanel(
    leafletOutput('mymap'))
) #ui

server <- function(input, output, session) {
  output$mymap <- renderLeaflet({
    km <- kmeans(selectedData, input$clusters)
    clustered_data <- data.frame(selectedData, km$cluster)

    leaflet(selectedData) %>%
      fitBounds(min(selectedData$data.Lon), min(selectedData$data.Lat),
                max(selectedData$data.Lon), max(selectedData$data.Lat)) %>%
      addTiles(group = "OSM (default)") %>%
      addProviderTiles("Stamen.Toner", group = "Toner") %>%
      addProviderTiles("Stamen.TonerLite", group = "Toner Lite") %>%
      addProviderTiles("OpenStreetMap.BlackAndWhite", group = "BlackandWhite") %>%
      addProviderTiles("CartoDB.DarkMatter", group = "Dark") %>%
      addCircleMarkers(~clustered_data$data.Lon, ~clustered_data$data.Lat,
                       radius = 7, color = ~pal(km$cluster), opacity=1) %>%
      addLayersControl(baseGroups = c("OSM (default)", "Toner", "Toner Lite",
                                       "BlackandWhite", "Dark"),
                       options = layersControlOptions(collapsed = FALSE)) %>%
      addLegend("bottomleft", pal=pal, values = clustered_data$km.cluster,
                title = "data by clusters", opacity = 1))
  } #server

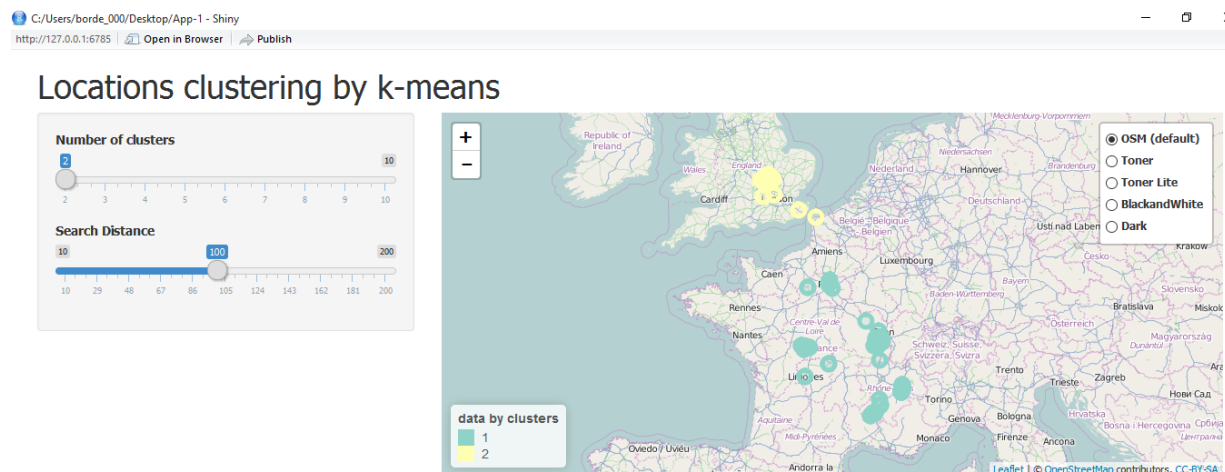
shinyApp(ui, server)
```

c. Results

There are 5 different basemaps in a menu, from which the user can choose (in the upper right corner). The Leaflet library supports basemaps using [map tiles](#). By default, [OpenStreetMap](#) tiles are used. Alternatively, there are free third-party basemaps that can be added, [here](#) is a set of possible basemaps. In addition to the default OpenStreetMap basemap, were chosen the followings 4 basemaps: "Stamen.Toner", "Stamen.TonerLite", "OpenStreetMap.BlackAndWhite" and "CartoDB.DarkMatter".

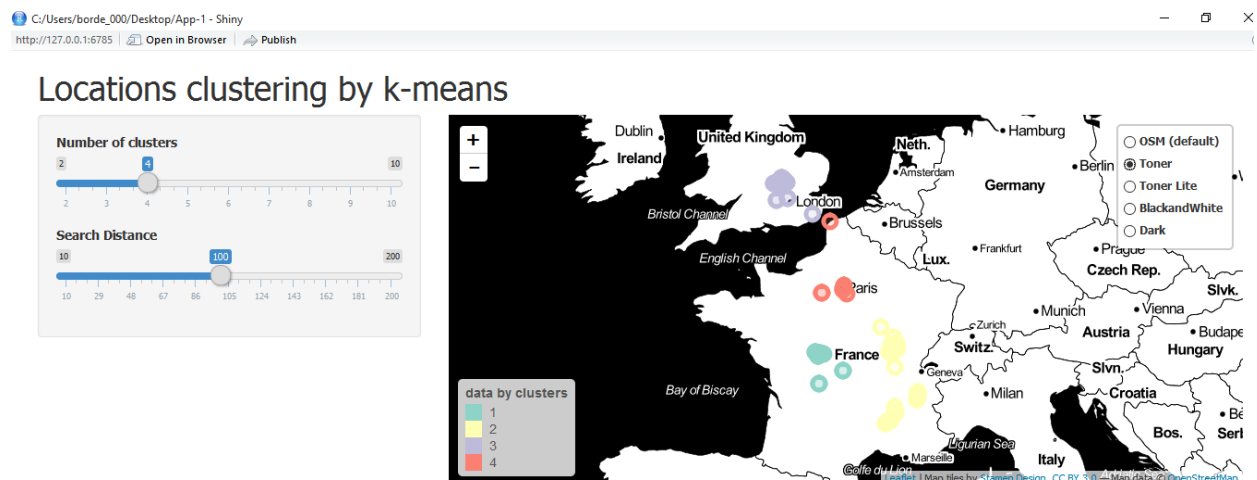
Here are the screen shots of this version:

- **Clustering results for k=2, the OSM default map is the base map:**

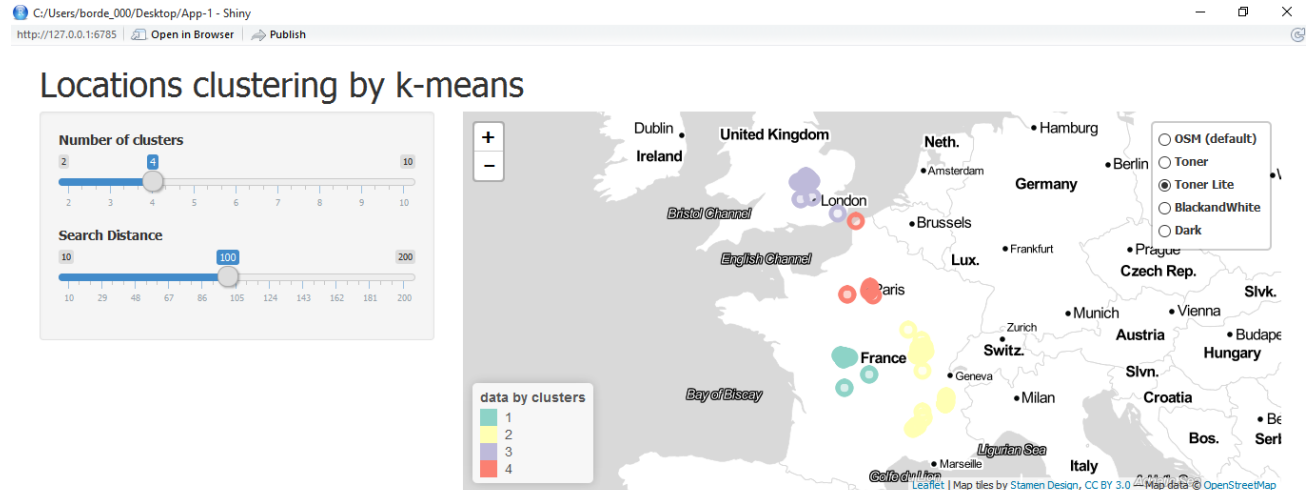


On the left part we can see the inputs required by the user. The first one is the value of k, which can be chosen to be some integer value in the range of 2-10. The second one is the “search distance” – at that point, it only appeared as an input slider, without actually doing something (was planned to be added later on with the rest of the code). What exactly this parameter will do is later explained.

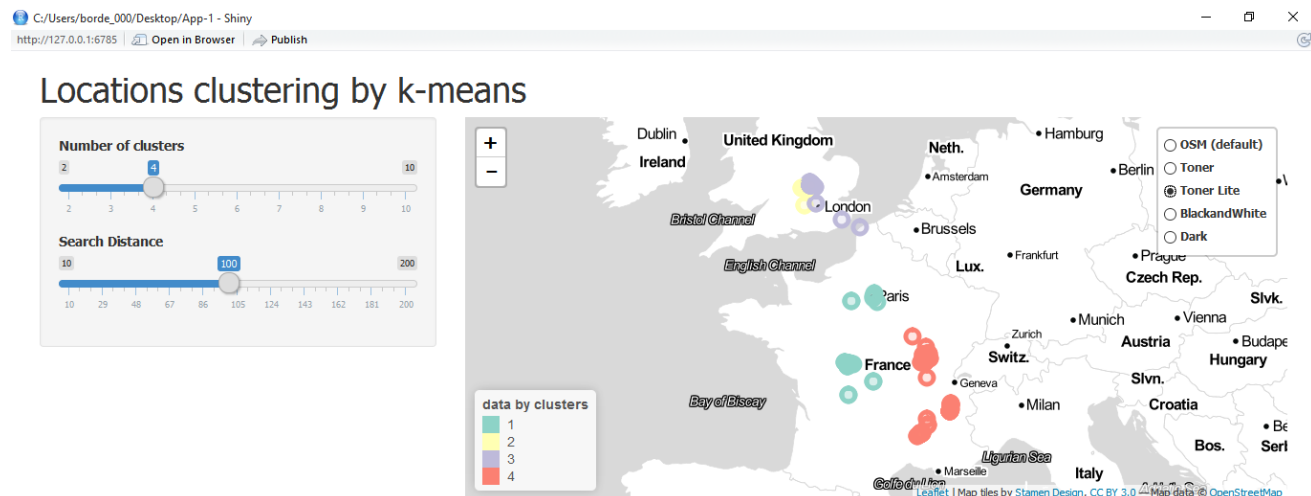
- **Clustering results for k=4, the Toner map is the base map:**



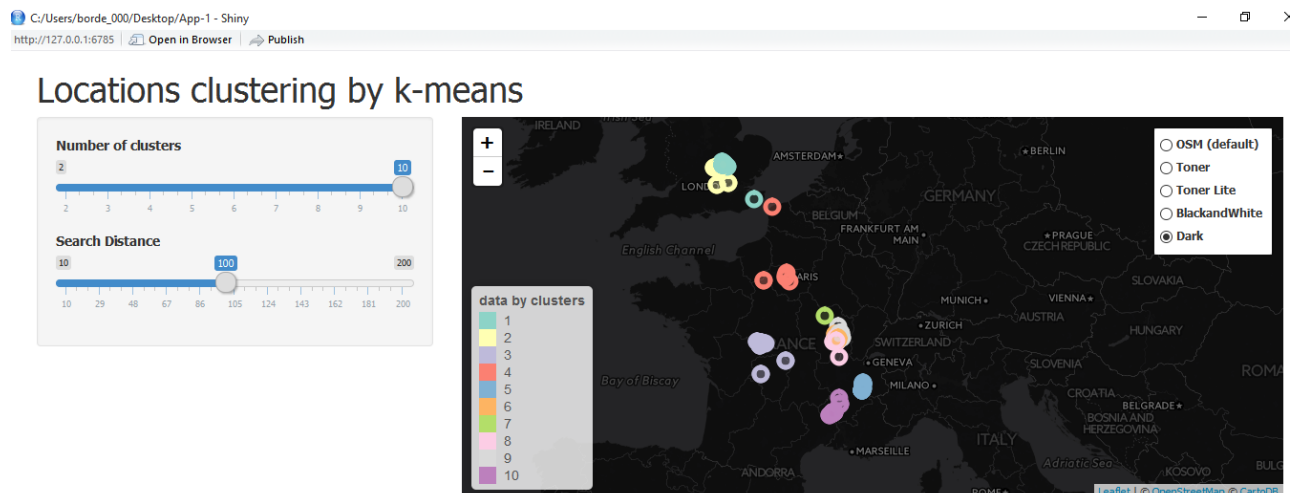
- Clustering results for $k=4$, the Toner Lite map is the base map:



- Clustering results for $k=4$, the BlackandWhite map is the base map:



- Clustering results for $k=10$, the Dark map is the base map:



d. Advantages / disadvantages / what is missing? /alternatives

- In this version there are a lot of things that are missing and were added in the next version (detailed below).
- Advantages:
 - The main one is the user interface and the interactivity which is provided by Shiny.
 - The opacity of the colors of the legend are set to 1, so it can be seen clearly even on the light basemaps.
- The disadvantages are:
 - Too many options of basemap which are not really needed. In the later version was decided to not to use the "Stamen.TonerLite" and "OpenStreetMap.BlackAndWhite" because they don't add much value to the visualization.
 - The color palette ("Set3") was chosen for the colors of the circle markers because it looks very good on the dark map but this colors are too light for the other basemaps and are sometimes difficult to notice. In order to overcome this problem, the opacity of the circles was set to 1, but it doesn't really solve this problem, so in the next version, the colors of the circles were changed to black and the fill color of them remained according to this palette.

➤ Second iteration: (final version)

The final version of the visualization, includes much more information but without the implementation in Shiny.

Regarding the base map, as already was mentioned, only three options were kept for the user to choose from, and their order in the menu were also changed. Now they are ordered by the order in which it will be more convenient (in my opinion) for the user to inspect the data.

a. Visual mapping of data-to-visual attributes:

The previous attributes remained the same:

- Geographical points (longitude, latitude) → location of circle markers on the map
 - Clustering results (to which cluster each data point belongs) → fill color of the circle marker
- The color of the circles is black and fill colors are according to the same palette ("Set3").

- Geographical size of the area which contains the data points of some cluster → rectangle around each cluster's points, which defines the "borders" of this area, in the same color as the fill color of the circle markers of that cluster.
- Semantic locations → the default blue markers of leaflet maps
This blue marker is added to each semantic location of type "node", and for the center of each semantic location of type (closed) "way"- which is a polygon. Each semantic polygon is also drawn on top of the map in the same color as the data in the corresponding cluster.
Clicking on the marker or on any point in the semantic polygon, opens a popup which says what location that is (the tags of the location). What is a "node" and what is a "way" is explained below.

In order to extract the semantic locations near each data point, in each cluster, was defined a "center_bbox" in which this point was the center point, and the height and width of the box (in meters) is defined by the value of the variable "search_dist" which is set by the user. In the following examples, this value was set to 100 meters.

Then, using the "osmar" library, in each "center_bbox" we search for the semantic locations. This library searches within the OpenStreetMap data. In OpenStreetMap data, the physical features on the ground (e.g., roads or buildings) are represented using tags attached to its basic data structures - its nodes, ways, and relations.

A node is a single point in space defined by its latitude, longitude and node id.

A way is an ordered list of nodes. A way can be open or closed. A closed way is one whose last node on the way is also the first on that way - may be interpreted either as a closed polyline, or an area, or both.

A relation is also an ordered list of one or more nodes, ways and/or relations as members which is used to define logical or geographic relationships between other elements. The "relation" data structure is less relevant in this case since we are interested in specific bounded locations. (https://wiki.openstreetmap.org/wiki/Map_Features)

Each tag describes a geographic attribute of the feature being shown by that physical feature. Therefore we are searching for data structures from type "node" and type "way" with specific tags. This tags were chosen from the tags list in the OpenStreetMap wiki page linked above

based on what seemed relevant for semantic locations of people's daily life (what exactly was chosen can be seen in the code).

- Summary of the semantic locations in each cluster → in the center of each rectangle which bounds the cluster, was added an icon (a cartoon image of a women sitting next to a desk). When it is clicked, a popup appears, in which is written the summary of the semantic data of the cluster – meaning, how many locations of each type there are in that cluster.

b. Code

```
library(RColorBrewer)
library(clusterSim)
library(leaflet)
library(osmar)
library(sp)
library(maptools)
library(rgdal)
library(rgeos)
library(htmltools)
library(plyr)
library(dplyr)
library(magrittr)

data <- read.csv("")
selectedData <- data.frame(data$Lon, data$Lat)
dataMatrix <- as.matrix(selectedData)

num_of_clusters <- 10
search_dist <- 100

km <- kmeans(selectedData, num_of_clusters)
clustered_data <- data.frame(selectedData, km$cluster)
semantic_data <- data.frame()

pal <- colorFactor(palette= brewer.pal(10,"Set3"), domain = c(1:10))
allDataMap <- leaflet(selectedData) %>%

fitBounds(min(clustered_data$data.Lon),min(clustered_data$data.Lat),

           max(clustered_data$data.Lon),max(clustered_data$data.Lat)) %>%

addProviderTiles("CartoDB.DarkMatter", group="Dark") %>%
addProviderTiles("Stamen.Toner", group = "Toner") %>%
addTiles(group = "OSM") %>%

addCircleMarkers(~selectedData$data.Lon, ~selectedData$data.Lat,
                 radius = 10, color = "black",fillColor= ~pal(km$cluster),
                 fillOpacity=0.5, group = "Data Points") %>%

addLayersControl(baseGroups = c("Dark", "Toner", "OSM"),
overlayGroups = c("Data Points", "Semantic Locations", "Semantics Summary"),
```



```

options = layersControlOptions(collapsed = FALSE)) %>%

addLegend("bottomleft",pal=pal,values = clustered_data$km.cluster,
          labels = clustered_data$km.cluster, title = "data by
clusters",opacity = 1)

allDataMap

# creating a table of the box values of each cluster
data_box_df <- data.frame()

#adding the bbox of each cluster
for (i in 1:num_of_clusters){
  ci_data <- filter(clustered_data, km.cluster==i)
  min_lon <- min(ci_data$data.Lon)
  min_lat <- min(ci_data$data.Lat)
  max_lon <- max(ci_data$data.Lon)
  max_lat <- max(ci_data$data.Lat)
  data_box <- corner_bbox(left=min_lon, bottom=min_lat, right=max_lon,
top=max_lat)

  data_box_df <- rbind(data_box_df,data_box)
  allDataMap <- allDataMap %>%
    addPolylines(c(data_box["left"],
                    data_box["right"],data_box["right"],
                    data_box["left"], data_box["left"]),
                  c(data_box["top"], data_box["top"],
                    data_box["bottom"],data_box["bottom"],
                    data_box["top"]),
                  col=(pal(x = i)), opacity = 1)
}
colnames(data_box_df) <- c("left", "bottom", "right","top")

src <- osmsource_api()

#####

for (cNum in 1:num_of_clusters){

  temp_c_data <- filter(clustered_data,km.cluster==cNum)

  for (j in 1:nrow(temp_c_data)){
    bb <- center_bbox(temp_c_data[j,1], temp_c_data[j,2],
                      search_dist,search_dist)

    ctown <- get_osm(bb, source = src)

    #####

    semanticNodes <- find(ctown,
                          node(tags((k == "amenity")|(k == "building")|
                                   (k == "emergency")| (k == "leisure")|
                                   (k == "office")| (k == "public_transport")|
                                   (k == "public transport")| (k == "shop")|
                                   (k == "sport")|(k == "tourism"))))

    semanticNodes <- find_down(ctown, node(semanticNodes))
    semanticNodes <- subset(ctown, ids = semanticNodes)
  }
}

```

semanticNodes

```
#check if there are any semantic nodes
if (nrow(semanticNodes$nodes$attrs) > 0) {
  semanticNodes_attr <- semanticNodes$nodes$attrs
  semanticNodes_tags <- semanticNodes$nodes$tags

  testLocs1 <- semanticNodes_tags$k == "amenity" |
    semanticNodes_tags$k == "building" |
    semanticNodes_tags$k == "emergency" |
    semanticNodes_tags$k == "leisure" |
    semanticNodes_tags$k == "office" |
    semanticNodes_tags$k == "public_transport" |
    semanticNodes_tags$k == "public transport" |
    semanticNodes_tags$k == "shop" |
    semanticNodes_tags$k == "sport" |
    semanticNodes_tags$k == "tourism"

  semanticNodes_tags <- semanticNodes_tags[testLocs1,]

  coords_table <- select(semanticNodes_attr, id, lon, lat)
  semanticNodes_tags_join <- semanticNodes_tags %>%
    left_join(coords_table, by = "id")

  #render the semantic nodes+popups
  allDataMap <- allDataMap %>%
    addMarkers(~semanticNodes_tags_join$lon,
              ~semanticNodes_tags_join$lat,
              group = "Semantic Locations",
              popup =
                ~htmlEscape(paste(
                  semanticNodes_tags_join$k, ":",
                  semanticNodes_tags_join$v))
            )

  temp_column <- matrix(data=rep(cNum, nrow(semanticNodes_tags_join)),
                        nrow = nrow(semanticNodes_tags_join), ncol=1)
  temp_column <- as.data.frame(temp_column)
  colnames(temp_column) <- c("cNum")
  temp_bind <- cbind(temp_column, semanticNodes_tags_join)
  semantic_data <- rbind(semantic_data, temp_bind)

} #if- semanticNodes

#####

semanticWays <- find(ctown, way(tags((k == "amenity") | (k == "building") |
  (k == "emergency") | (k == "leisure") |
  (k == "office") | (k == "public_transport") |
  (k == "public transport") |
  (k == "shop") | (k == "sport") |
  (k == "tourism")
)))

semanticWays <- find_down(ctown, way(semanticWays))
```

```

semanticWays <- subset(ctown, ids = semanticWays)
semanticWays

if (nrow(semanticWays$ways$attrs) > 0){
  semantic_ways_poly <- as_sp(semanticWays,"polygons")
  ways_bbox <- bbox(semantic_ways_poly) #the bbox surrounding all polygons

  semanticWays_tags_join <- data.frame()

  # To add popups for polygons
  semanticWays_ways_attrs <- semanticWays$ways$attrs
  semanticWays_ways_tags <- semanticWays$ways$tags

  testLocs2 <- semanticWays_ways_tags$k == "amenity" |
    semanticWays_ways_tags$k == "building" |
    semanticWays_ways_tags$k == "emergency" |
    semanticWays_ways_tags$k == "leisure" |
    semanticWays_ways_tags$k == "office" |
    semanticWays_ways_tags$k == "public_transport" |
    semanticWays_ways_tags$k == "public transport" |
    semanticWays_ways_tags$k == "shop" |
    semanticWays_ways_tags$k == "sport" |
    semanticWays_ways_tags$k == "tourism"

  semanticWays_ways_tags <- semanticWays_ways_tags[testLocs2,]

  #extracting from all polygons, jusy the specified semantic ones
  for(i in 1:length(semantic_ways_poly@polygons)){

    poly1 <- (semantic_ways_poly@polygons[i]) #list of polygons
    poly1l <- poly1[[1]]@Polygons #list of 1 polygon
    poly1l_id <- poly1[[1]]@ID
    amenity_V_index <- match(poly1l_id,
      semanticWays_ways_tags$id,
      nomatch=0) #the positions of poly1l in the tags data

    if (amenity_V_index != 0){
      poly1.coords <- poly1l[[1]]@coords

      allDataMap <- allDataMap %>%
        addPolygons(lng = poly1.coords[,1],
          lat = poly1.coords[,2],
          group = "Semantic Locations",
          color = (pal(x = cNum)), opacity = 0.5,
          popup =
            paste(semanticWays_ways_tags[amenity_V_index,"k"],
              ":",
              semanticWays_ways_tags[amenity_V_index,"v"]))
        %>%
        addMarkers(lng = mean(poly1.coords[,1]),
          lat = mean(poly1.coords[,2]),
          group = "Semantic Locations",
          popup = paste("Center of:",
            semanticWays_ways_tags[amenity_V_index,"k"],
            ":",
            semanticWays_ways_tags[amenity_V_index,"v"]))

    }

  }

  mean_coords <- data.frame(lon=mean(poly1.coords[,1]),

```

```

lat = mean(poly1.coords[,2]))
semanticWays_tags_temp <- cbind(semanticWays_ways_tags[amenity_V_index,],
                                mean_coords)
semanticWays_tags_join <- rbind(semanticWays_tags_join, semanticWays_tags_temp)
} #if
} #for

temp_column2 <- matrix(data=rep(cNum, nrow(semanticWays_tags_join)),
                       nrow = nrow(semanticWays_tags_join), ncol=1)
temp_column2 <- as.data.frame(temp_column2)
colnames(temp_column2) <- c("cNum")
temp_bind2 <- cbind(temp_column2, semanticWays_tags_join)
semantic_data <- rbind(semantic_data, temp_bind2)

} #if - semanticWays

} # for - all points in the cluster

} #for- cNum

#don't want to count the same semantic location in each cluster several times
unique_data <- unique(semantic_data)
row.names(unique_data) <- NULL
unique_semantic_data <- select(unique_data, cNum, k, v)
count_summary <- ddply(unique_semantic_data, .(cNum, k, v), nrow)

leafIcons <- icons(iconUrl =
"http://icons.iconarchive.com/icons/dapino/office-women/48/eyes-office-women-
glasses-icon.png",
iconWidth = 38, iconHeight = 95)

allDataMap2 <- allDataMap

for (i in 1:num_of_clusters) {
  summary <- filter(count_summary, cNum==i)
  summary <- select(summary, k,v, V1)
  content <- ""

  for (j in 1:nrow(summary)){
    t <- paste(summary[j,1], summary[j,2], ": ", summary[j,3])
    content <- paste(sep = "<br/>",
                    content,
                    t)
  }

  lon <- mean(c(data_box_df[i,"left"], data_box_df[i,"right"]))
  lat <- mean(c(data_box_df[i,"bottom"], data_box_df[i,"top"]))

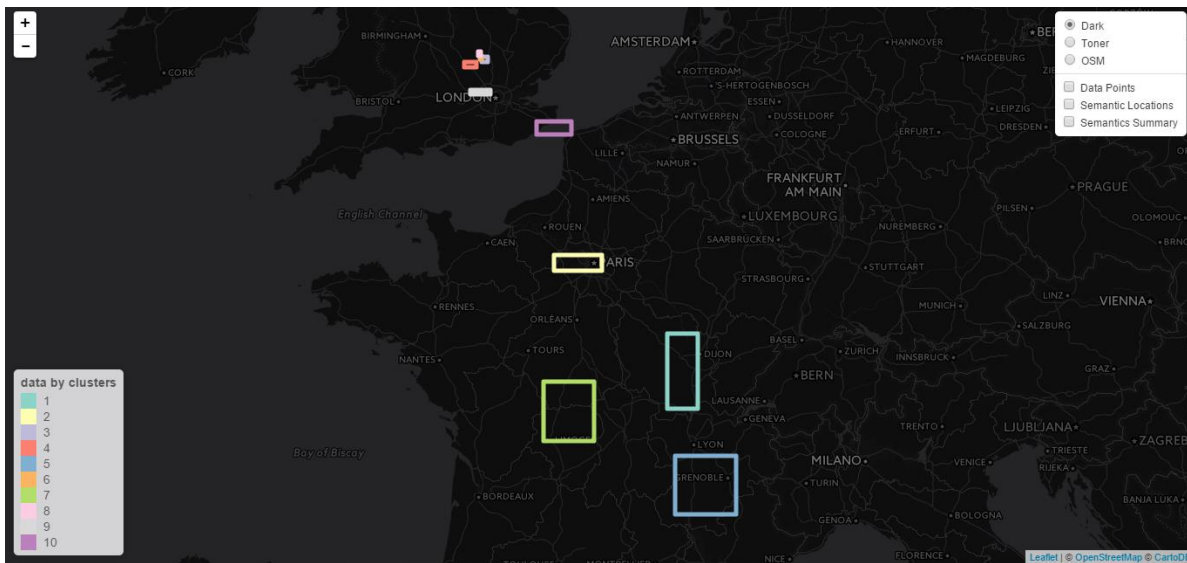
  allDataMap2 <- allDataMap2 %>% addMarkers(lng = lon, lat = lat,
                                           icon=leafIcons, group="Semantics
Summary",
                                           popup = content
)
}
allDataMap2

```

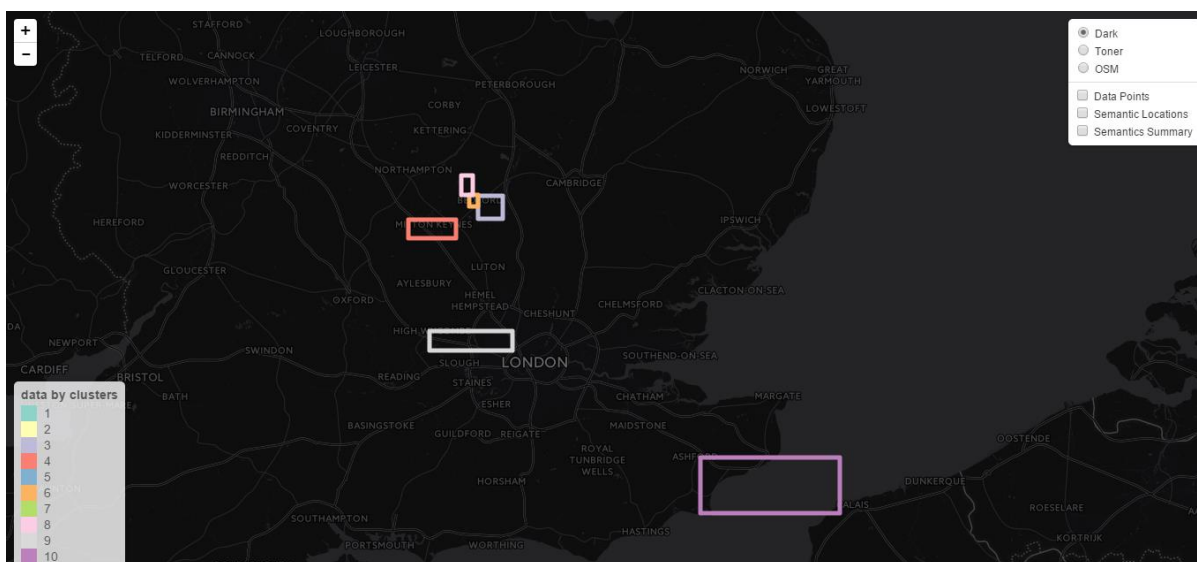
c. Results

The output can be seen with one of three possible base maps, and on top of that, the user can choose the information he wants to see on the map – the data points (circle markers) and/or the semantic locations (blue markers + polygons) and/or the summary of the semantic locations of each cluster. The rectangles around each clusters points are always seen, so that it can always be seen where the clusters are located and how big is the area they cover, regardless of the additional information the user can choose to see.

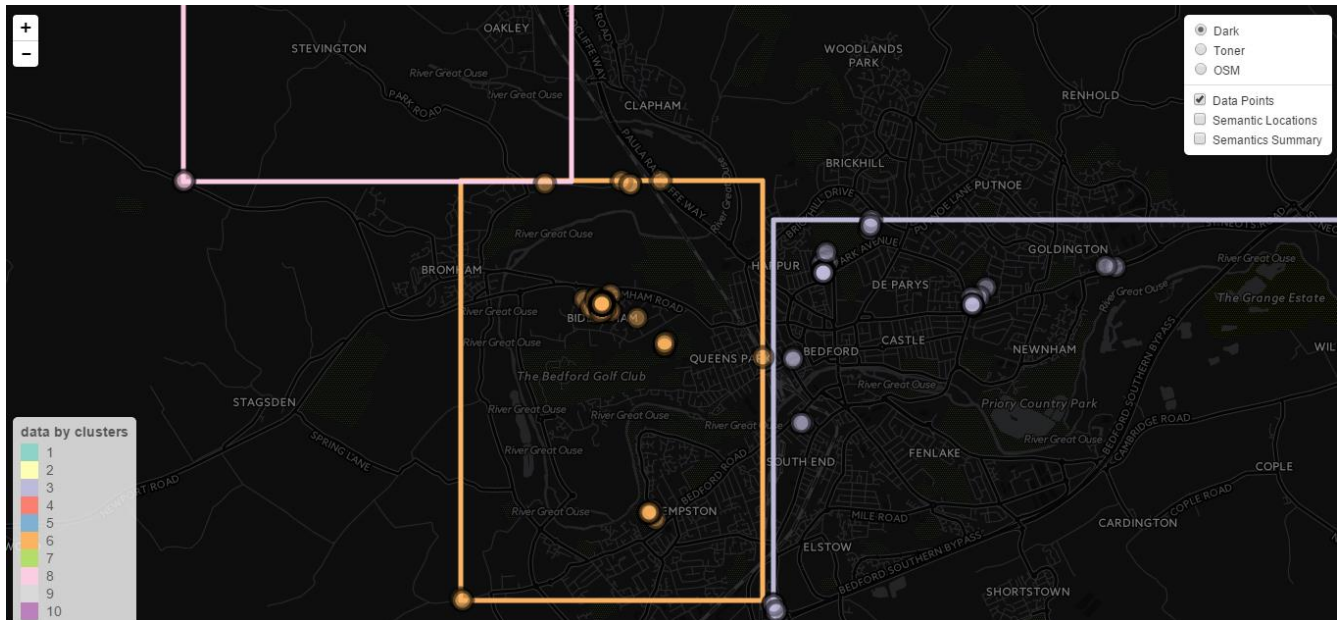
- The rectangles around each clusters points, which defines the “borders” of the area:



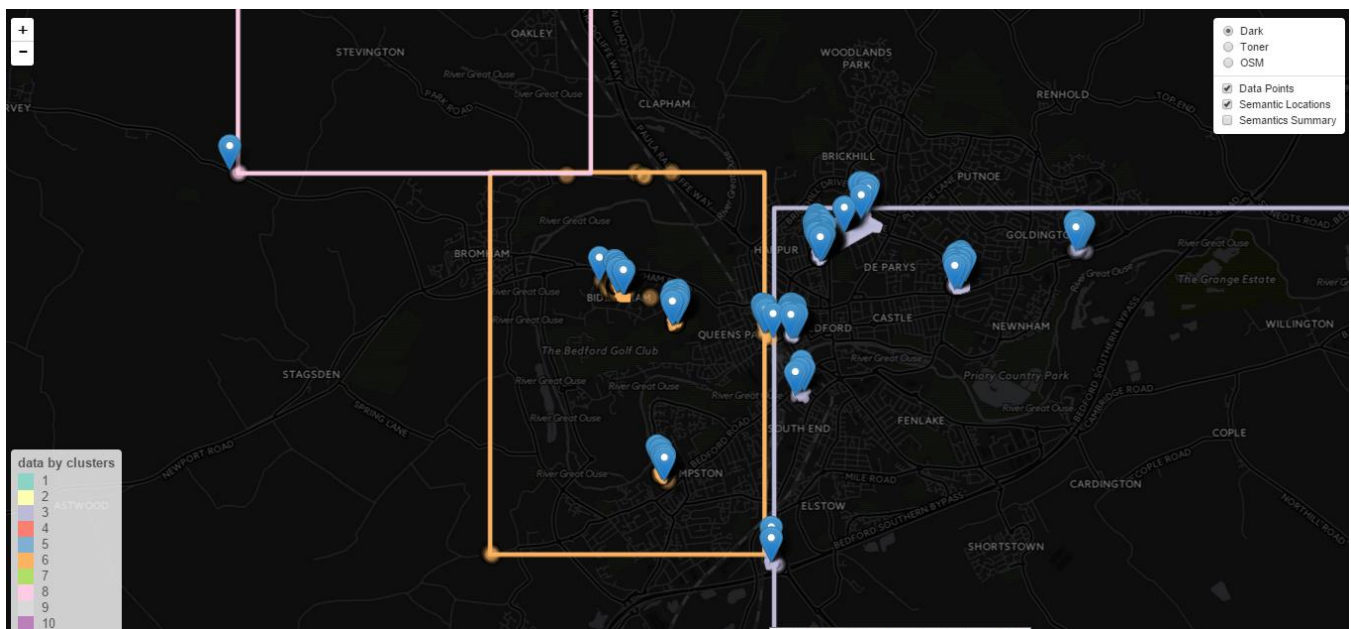
- Zooming in to the data in UK:



- Adding the data points and looking for example at what's in the orange cluster:

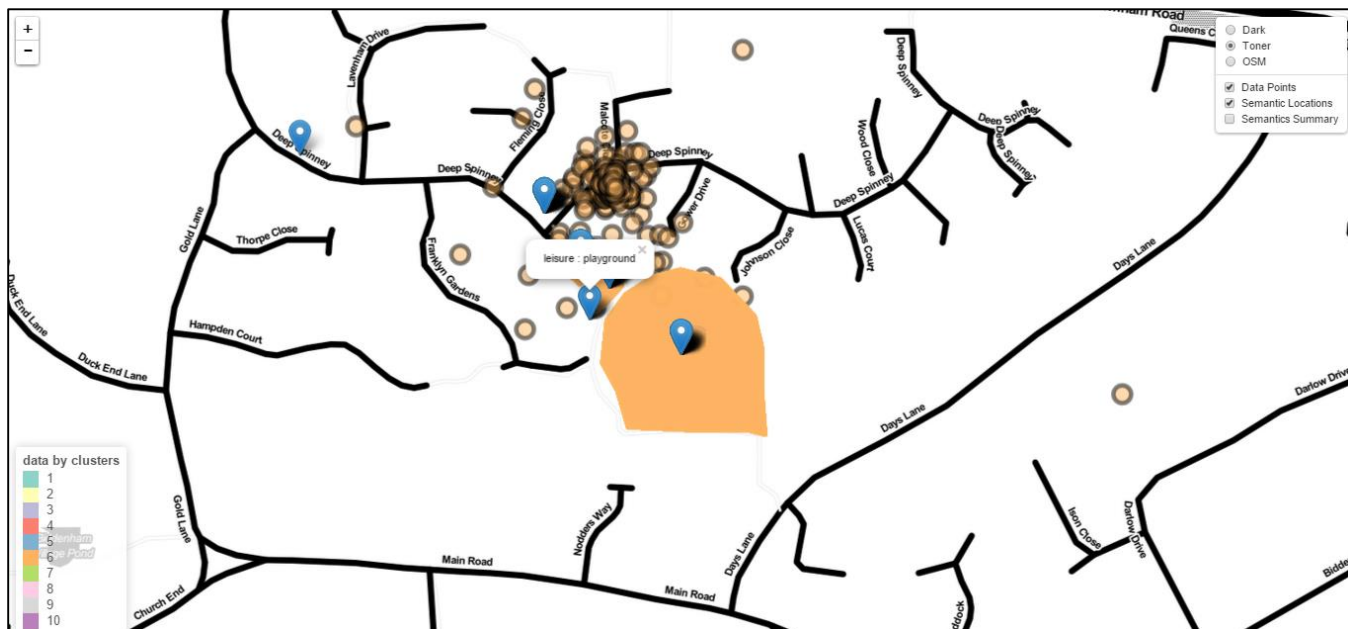


By adding the layer of semantic information we can see that the orange cluster has 4 semantic location concentrations, three in the middle and one on the right border:



- **Zooming in and looking at the semantics:**

Here we can see many data points near a pitch and a playground:

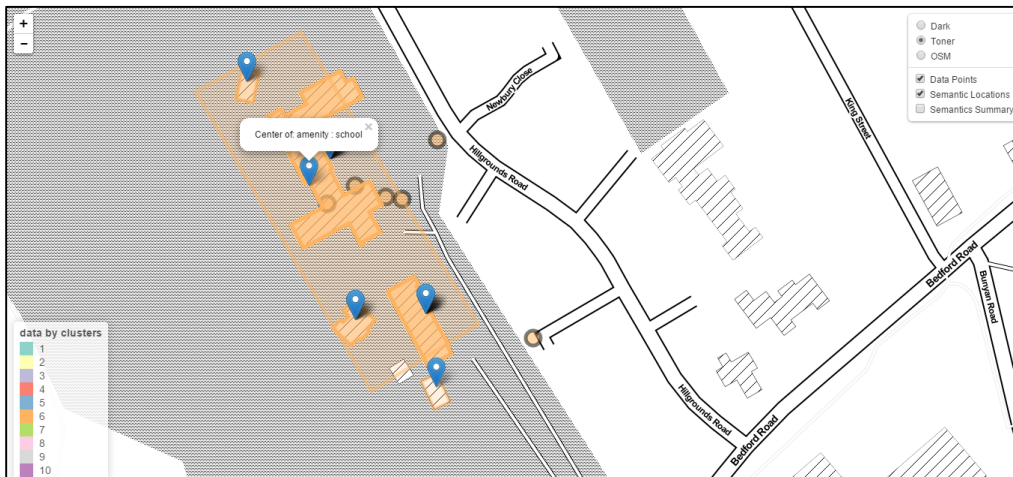


The area of the pitch is the big orange polygon. All polygons opacity is set to 0.5, but this polygon seems dark. This means the same polygon was drawn several times because it's a semantic location which was found to be near many data points. A more transparent polygon will indicate a location which was found to be close to less data points.

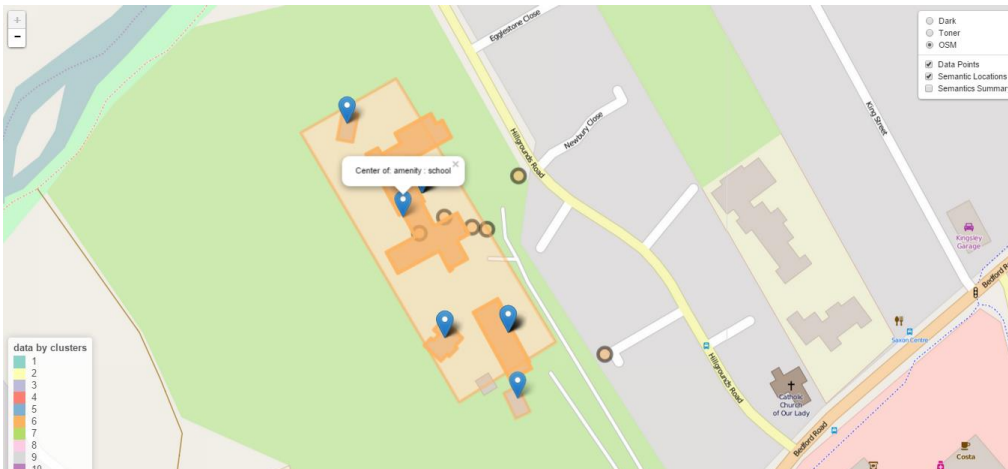
Another concentration of points is near a school:



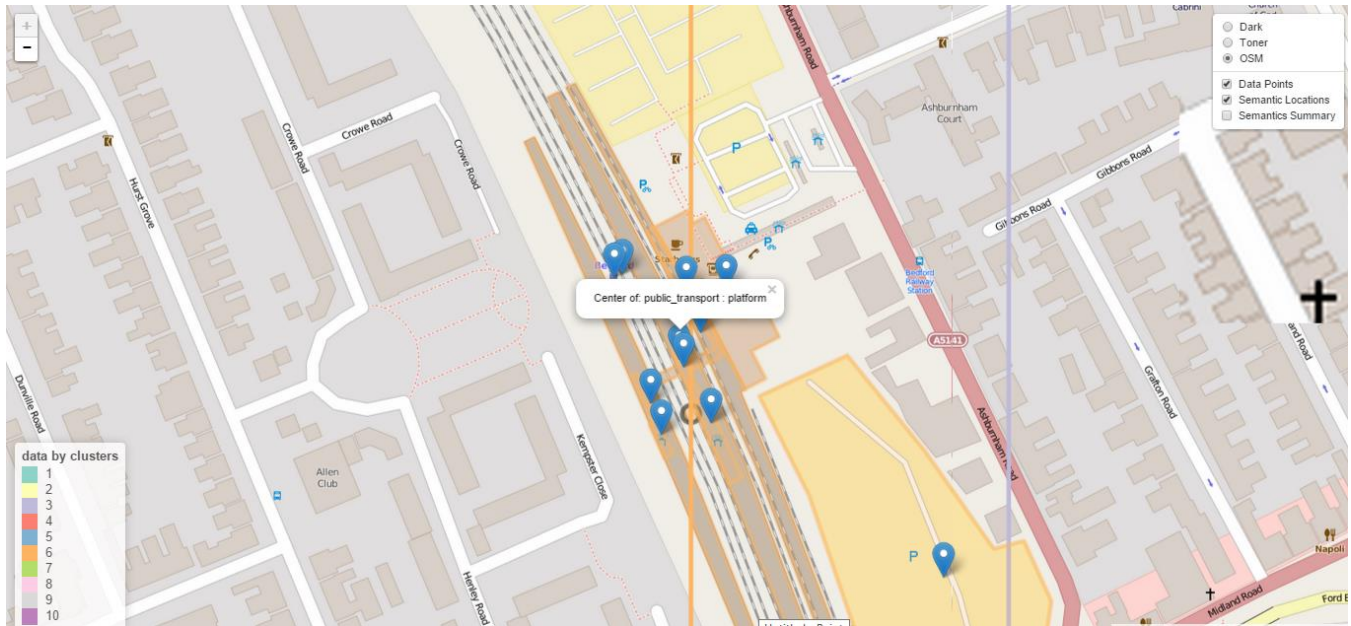
Another concentration near another school:



Same area with an OSM base map (we can see that the gray area around the school is just lawn):

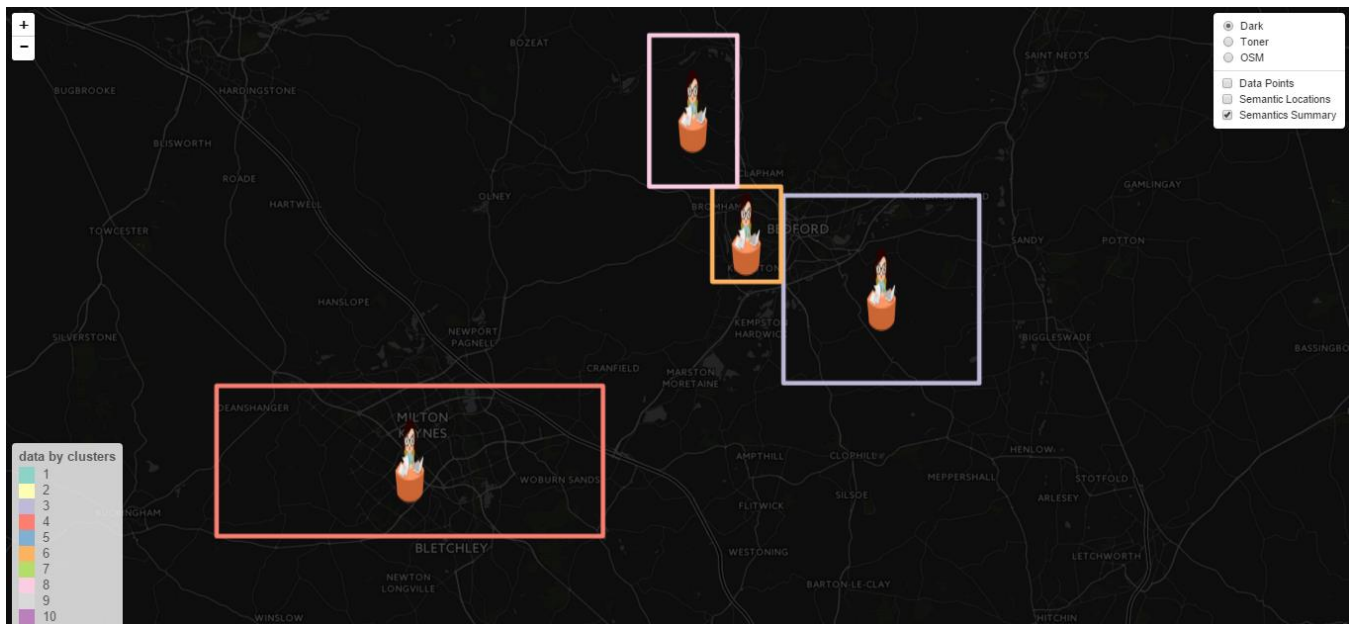


And the fourth concentration of semantic points is near a public transport platform:

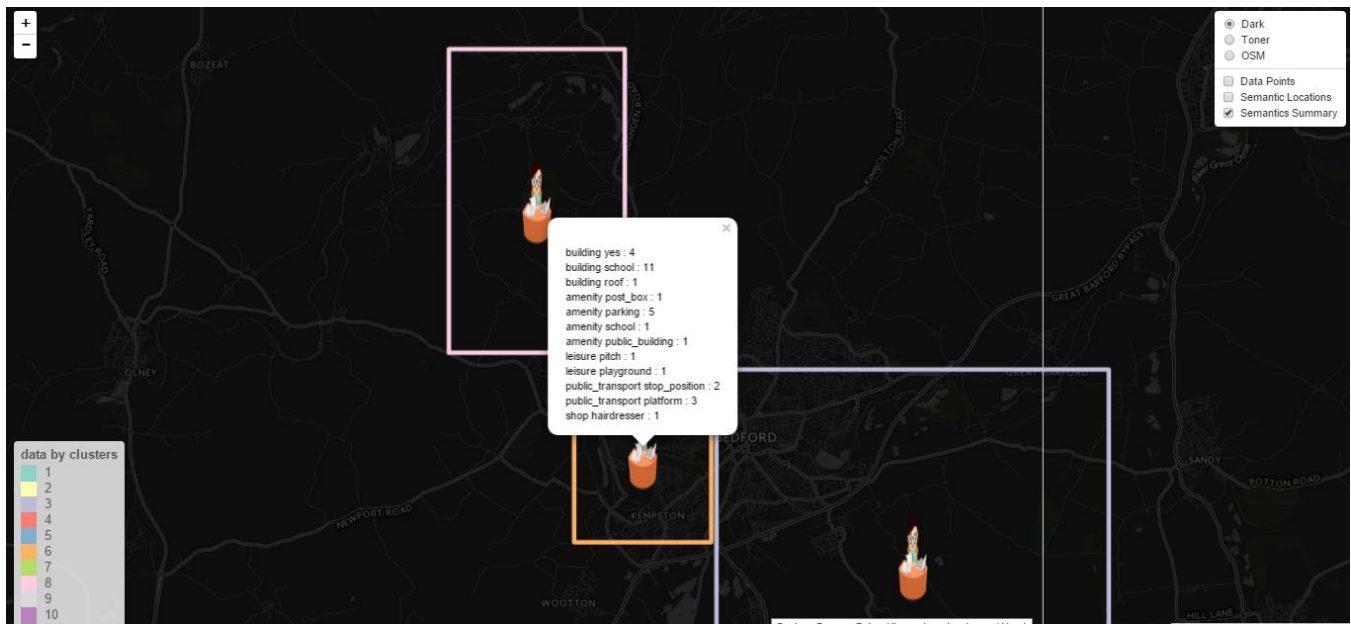


- **Semantic summary per cluster:**

Finally we can see the summary of everything there is in the orange cluster - when the “Semantic Summary” option is chosen (on the menu on the right), an icon appears in the middle of each cluster:



Clicking on the icon makes a popup with the summary appear:





The meaning of the summary:

For example, “Building school: 11” means that 11 different locations with the tag “Building school” (key = “Building”, value=“school”) were found near the data points of the orange cluster.

d. Advantages / disadvantages / what is missing? /alternatives

- Advantages:
 - The possibility to switch between different basemaps – allows us to look at the data from different points of view. For example, at the beginning if we just want to see where geographically the data is located, how big/small are the clusters and how close/far they are from each other, without additional visual “noise”, it is convenient to use the Dark map, if we want to see more details, it will be more convenient to use the other maps.
 - Not all of the data has to be seen at once - the user can choose what information he wants to see on the map (data points/semantic locations/semantic summary)
 - Circle markers opacity set to 0.5– allows us to see where there are many data points overlying each other and where there are not.
 - The opacity of the semantic polygons is set to 0.5 so every time a semantic location is found to be close to another data point we can tell the user visited somewhere near that location many times, so it is more likely that is the actual place he is visiting.
 - The blue markers which denote the semantic locations (location of type “node” or the center of a semantic polygon) can be seen well at different zooming levels and on top of all the basemaps.
 - The visualization output – this interactive map, can be saved as an HTML page and reproached without the need to re run the code.
- Disadvantages:
 - The user needs to help himself to better understand what he sees on the map by zooming in and zooming out, changing the basemaps and choosing what information he wants to be displayed on the map.
 - When the clusters are far from each other, e.g. there are data points in different countries, it is difficult to notice the smaller ones – we need to zoom in to see it better.
 - When several circle markers overlay each other, it’s difficult to read the names of the countries/cities etc. on the map.
 - Same blue markers are used to denote all the semantic locations of all clusters – it doesn’t allow us to compare between different types of semantic locations within the same cluster or see the differences between the semantic locations of different clusters, based on this markers..

- The summary of the semantic locations which were found near the data points of each cluster, appears as a list in a popup, it is difficult to compare between the summaries of all clusters.
- Not all the extracted semantic data is really interesting for the purposes of this visualization, for example, semantic tags such as parking or just a building, is not really necessary.
- The k-means clustering algorithm is not the best one for clustering spatial data. One reason for that is that in this algorithm there will not be “noise” points (unlike the DBSCAN algorithm for example), meaning, every data point will be assigned to some cluster, which is not necessarily a good thing in this case.
- What’s missing and alternatives:
 - Missing the complete implementation in Shiny.
 - Finding better markers for the semantic locations – e.g. for all locations tagged as shop, assign a dedicated marker icon, like this  , and for a park, an icon like this .
 - Filtering more tags which define what of locations to extract – e.g. omitting tags like “building: yes”.
 - Finding a better way to present the summary of the semantic locations of each cluster in the popup – e.g. in a form of a table containing only the meaningful semantic tags.

4. Evaluation

a. Reflection on design decisions

- Keeping the Toner map – it is a good basemap in my opinion, not sure if it is really necessary.
- The chosen color palette (“Set3” of color brewer) is very good for the Dark basemap but less good for the other basemaps. Perhaps it was better to use a darker palette.
- Deciding not to assign the k parameter of the k-means algorithm a value higher than 10: although rendering to many clusters and thus to many colors on the map may be too much to capture, even 10 clusters may not be enough to fully divide into groups the locations a person visits, and it will still be in a too high resolution, as we could see in the presented demonstration.

In this kind of situation, it will be needed to re run everything just on the data points belonging to a certain cluster, to receive the partition of locations in a lower resolution.

b. Assessment of the Value of Visualization

- The user will be able to see how the results will change as a function of the value of k and also as a function of the value of “search_dist”.
- This visualization is providing the user a tool for the assessment of the output of his clustering algorithm. It does require the user to be active and explore things in an interactive way, but it is very helpful to be able to see on top of a map where the data is located, what are the relevant semantic locations near the data points etc.
- Even though there are quantitative measures for clustering evaluation, like Dunn Index or Davis-Bouldin Index, which take into consideration the density of the points within each cluster and how far the clusters are from each other yet in this case, they may not be sufficient to find the optimal clustering. For example, let’s assume that some user has many data points but they are all close only to three semantic locations, one is a University (assumingly his work place) and the two others are his home and his kid’s school. Let’s also assume all of these locations are far from each other and that the university is a geographically very large area comparing to the home and the school, and that the data points related to the university are spread in different locations within the university.

So it is likely that for a parameter of $k=5$ for example, we will see that all points near the home are grouped into one cluster, all points near the school are grouped in another cluster and the points corresponding to the university area are separated into 3 other clusters. If we will set k to be 3 we will probably see the “correct” partition to clusters, but it is very likely that if we need to choose the better outcome (meaning to choose the better result between $k=3$ or $k=5$) based on Dunn Index, we will see that it “prefers” the option of $k=5$ because it can only take into consideration the “geography” and not the semantics. So the main goal of this visualization project is to help the researcher to evaluate the clustering outcome based on the semantic data.