Project - spatial clustering of point data

Katarzyna Tokarczuk

2025-02-05

Loading Files and Required Libraries

```
library(sf)
library(ggplot2)
library(dbscan)
library(dplyr)
library(RColorBrewer)
points <- st_read("C:/Users/kasia/Studies/SEMESTR 5/Analiza danych przestrzennych/projekt/zestaw8_XYTabl
neighborhoods <- st_read("C:/Users/kasia/Studies/SEMESTR 5/Analiza danych przestrzennych/projekt/osiedla
```

1. DBSCAN Method

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm

Working Principle:
DBSCAN is a data clustering algorithm based on analyzing the relative positions and distances between points, meaning it relies on spatial density.

Algorithm Steps:
1. Calculating the neighborhood - for each point in the dataset, its neighborhood is determined as the points within a radius of eps from that point.
2. Identifying core points - a point is considered a core point if the number of points in its neighborhood is greater than or equal to minPts.
3. Connecting core points into clusters - core points that are density-reachable are grouped into clusters.
4. Assigning border points - border points, which are not core points but are within the neighborhood of core points, are assigned to the clusters formed by those core points.
5. Assigning noise points - points that are neither core nor border points are marked as noise (cluster 0).

Disadvantages:
- It struggles to cluster datasets with significant differences in density (since it is difficult to find a single combination of minPts and eps that works for all clusters).
- The quality of the algorithm depends on the chosen distance metric.
- It is not entirely deterministic.

Advantages:
- No need to specify the number of clusters.
- Handles noise and outliers well.
- Requires only two parameters.

I use the dbscan function and set different parameter values for this method:
eps - the radius within which neighbors are counted.
minPts - the minimum number of points required to form a cluster.

```r
# Extracting coordinates from the point file
coords <- st_coordinates(points)
head(coords)
```

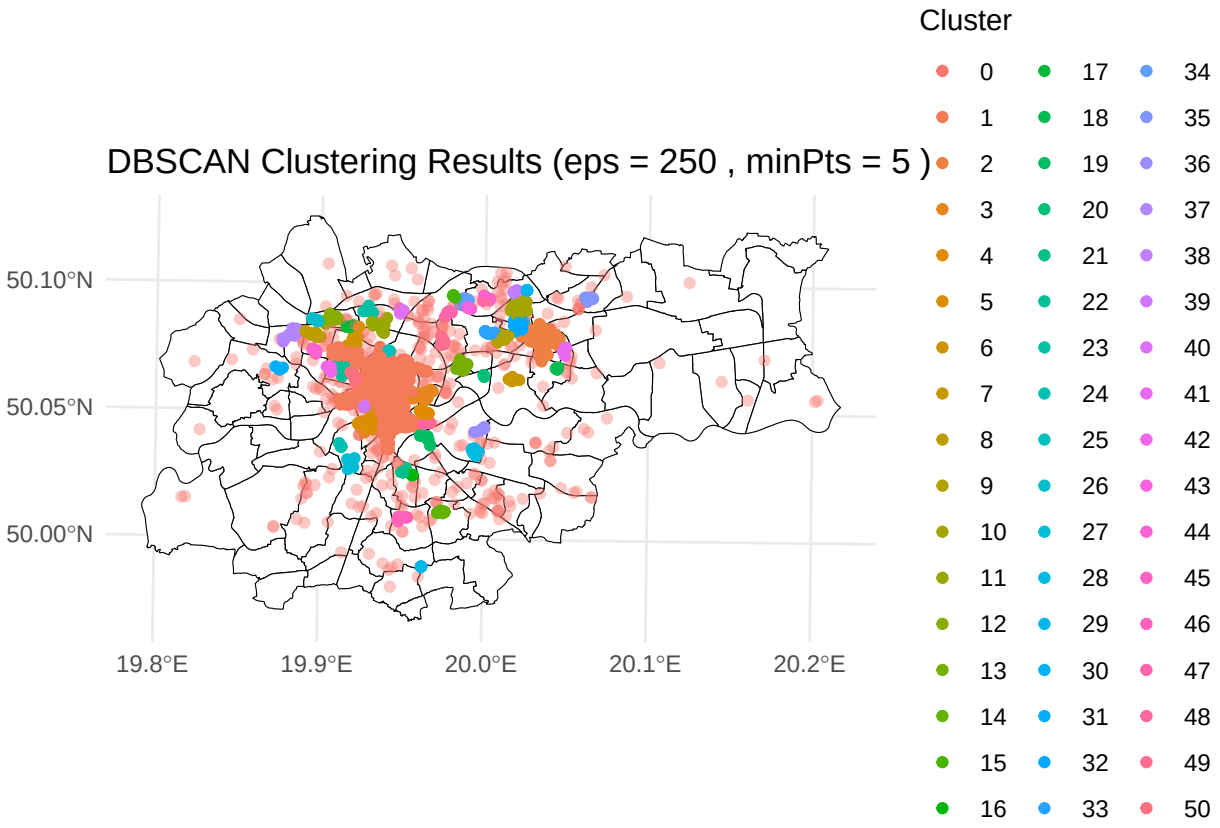
```
            X       Y
  [1,] 7423881 5547947
  [2,] 7430823 5549041
  [3,] 7425816 5547329
  [4,] 7423685 5548121
  [5,] 7423867 5547753
  [6,] 7429003 5551706
```

```r
# Number of points
coords_count <- nrow(coords)
print(coords_count)
```

```
  [1] 2000
```

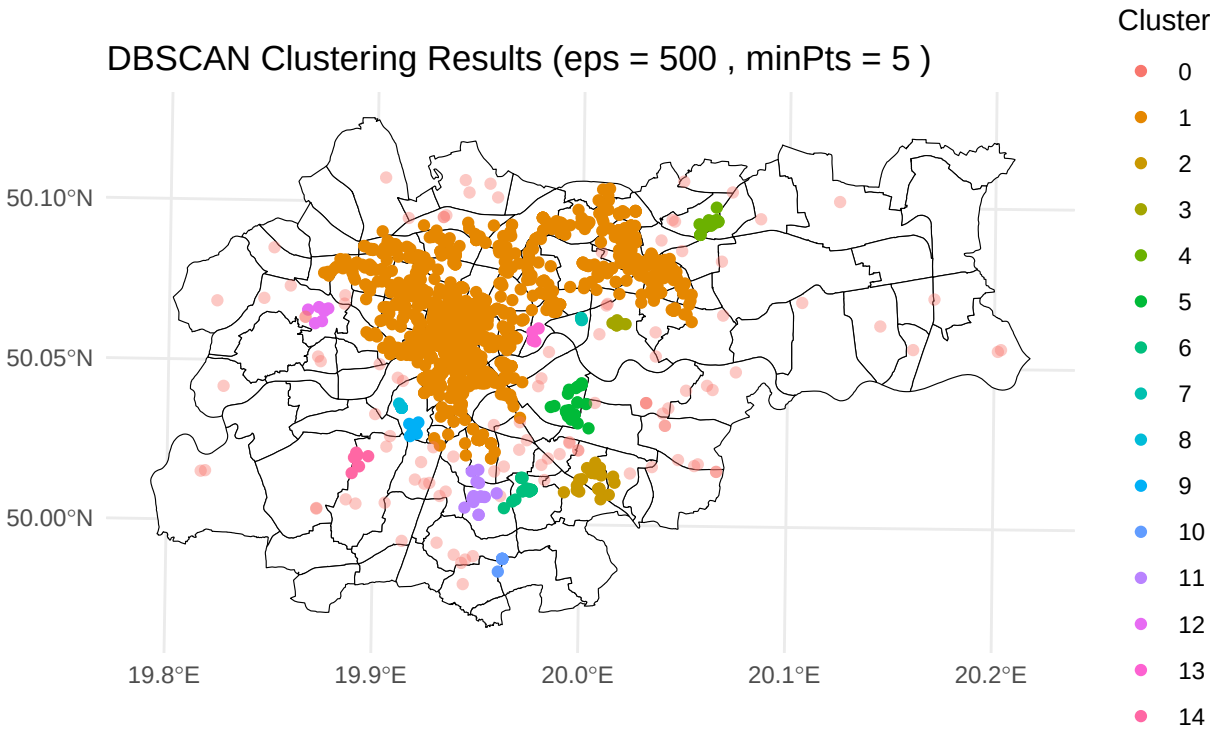
```r
# Helper function for DBSCAN clustering and visualization
clustering_dbscan <- function(data_points, data_neighborhoods, eps, minPts) {
  coords <- st_coordinates(data_points)
  db <- dbscan(coords, eps = eps, minPts = minPts)
  data_points$cluster <- db$cluster

  ggplot() +
    geom_sf(data = data_neighborhoods, fill = "white", color = "black") +
    geom_sf(data = data_points, aes(color = as.factor(cluster), alpha = ifelse(cluster == 0, 0.4, 1))) +
    theme_minimal() +
    labs(color = "Cluster", alpha = "Transparency") +
    ggtitle(paste("DBSCAN Clustering Results (eps =", eps, ", minPts =", minPts, ")")) +
    scale_alpha_identity()
}
```

```r
# DBSCAN Clustering (eps=250, minPts=5)
clustering_dbscan(points, neighborhoods, eps = 250, minPts = 5)
```



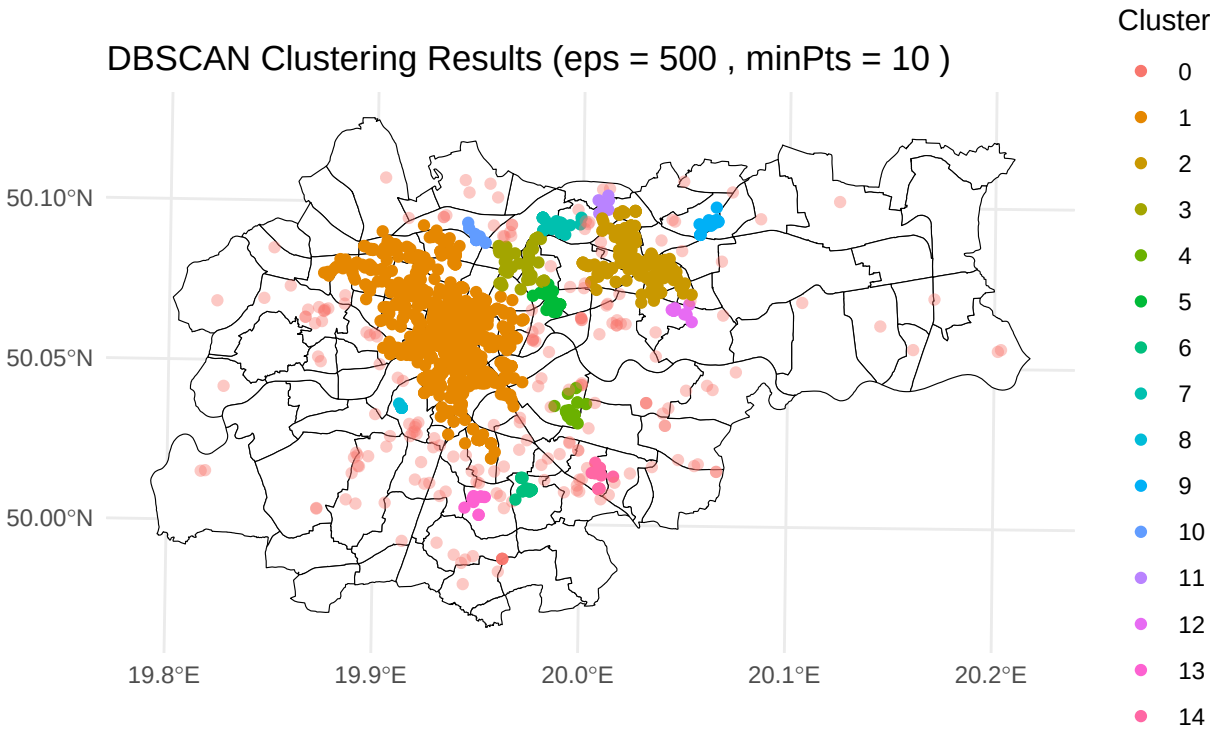
The minPts value of 5 with a neighborhood radius of 250 resulted in a large number of points being assigned to many small clusters.

```
# DBSCAN Clustering (eps=500, minPts=5)
clustering_dbscan(points, neighborhoods, eps = 500, minPts = 5)
```



In this case, the eps parameter was increased to 500, while minPts remained at 5. A total of 14 clusters were formed.

```
# DBSCAN Clustering (eps=500, minPts=10)
clustering_dbscan(points, neighborhoods, eps = 500, minPts = 10)
```



In the next case, eps remained at 500, while the minimum number of points required to form a cluster was increased. These data provide more insights into the increased intensity of offenses.

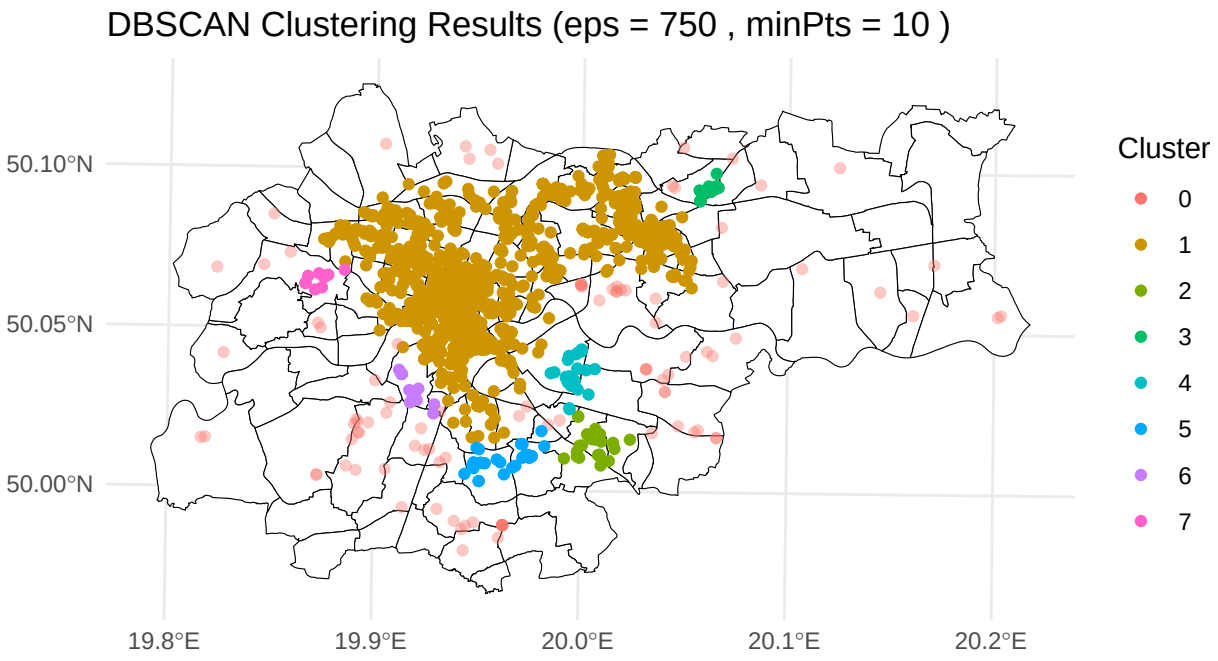
```
# Function to summarize clusters by neighborhoods
cluster_summary <- function(data_points, data_neighborhoods, eps, minPts) {
  coords <- st_coordinates(data_points)
  db <- dbscan(coords, eps = eps, minPts = minPts)
  data_points$cluster <- db$cluster

  clustered_points <- data_points %>% filter(cluster != 0)
  points_with_neighborhoods <- st_join(clustered_points, data_neighborhoods, join = st_within)

  result <- points_with_neighborhoods %>%
    group_by(cluster, NAZWA_JEDN) %>%
    summarise(num_violations = n(), .groups = "drop") %>%
    st_drop_geometry() %>%
    arrange(desc(num_violations))

  head(result)
}
cluster_summary(points, neighborhoods, eps = 500, minPts = 10)
```

```
  # A tibble: 6 x 3
    cluster NAZWA_JEDN          num_violations
      <int> <chr>                        <int>
  1       1 "Stare Miasto"                 404
```

```
2       1 "Kazimierz"                   169
3       1 "Warszawskie"                  83
4       1 "Nowy \x8cwiat"                64
5       1 "Stare Podg\xf3rze"            60
6       1 "Kleparz"                      59
```

Above is the result of the auxiliary function. The presence of clusters indicates that the areas with increased intensity of recorded offenses are: Stare Miasto, Kazimierz, Warszawskie, Nowy Świat, Stare Podgórze, and Kleparz.

```
# DBSCAN Clustering (eps=750, minPts=10)
clustering_dbscan(points, neighborhoods, eps = 750, minPts = 10)
```



When minPts remained at 10 and the eps radius was increased to 750, the number of clusters decreased.

---

2. HDBSCAN Method

The HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm

Working Principle:
HDBSCAN is a hierarchical spatial clustering algorithm that identifies clusters based on density while accounting for noise. It defines cluster cores as points that have a sufficient number of neighboring points

within a specified threshold. Points that are within a close distance of any core point (closer than a given value) are then added to the cluster, forming a complete cluster structure.

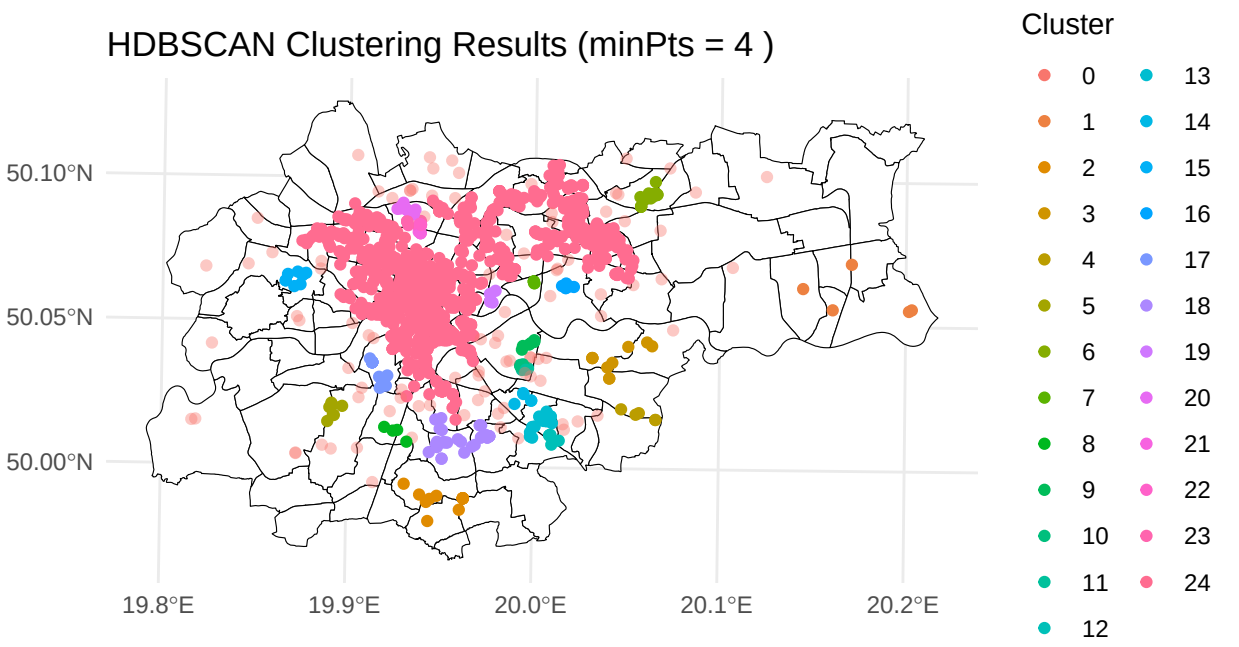
Algorithm Steps: **1.** Compute mutual reachability distance.
2. Construct a minimum spanning tree.
3. Trim the tree based on stability.
4. Extract clusters.

Disadvantages:
- Sensitive to sparse data.
- Slower than DBSCAN for large datasets.

Advantages:
- Handles clusters with varying density.
- Automatically selects parameters.
- Produces stable clusters.

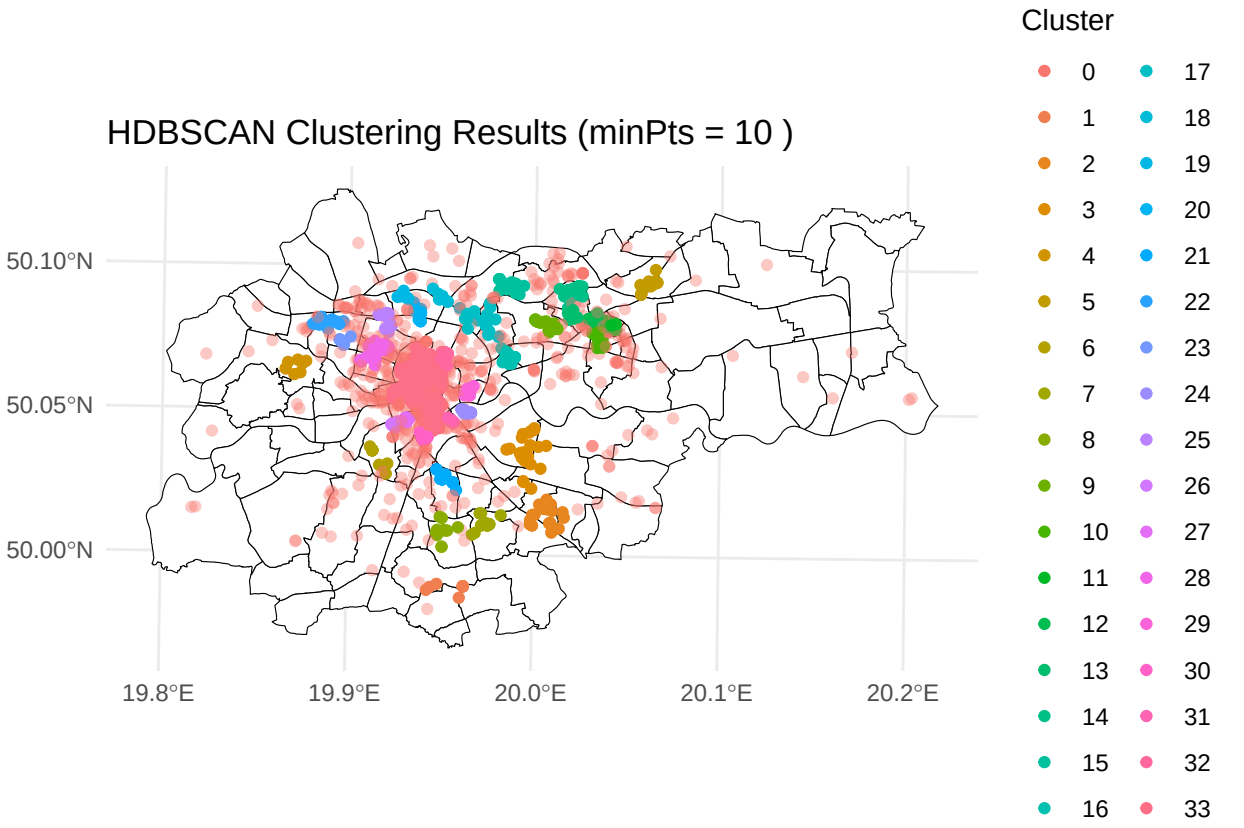
```r
# Helper function for HDBSCAN clustering and visualization
clustering_hdbscan <- function(data_points, data_neighborhoods, minPts) {
  coords <- st_coordinates(data_points)
  db <- hdbscan(coords, minPts = minPts)
  data_points$cluster <- db$cluster

  ggplot() +
    geom_sf(data = data_neighborhoods, fill = "white", color = "black") +
    geom_sf(data = data_points, aes(color = as.factor(cluster), alpha = ifelse(cluster == 0, 0.4, 1))) +
    theme_minimal() +
    labs(color = "Cluster", alpha = "Transparency") +
    ggtitle(paste("HDBSCAN Clustering Results (minPts =", minPts, ")")) +
    scale_alpha_identity()
}
```

```r
# HDBSCAN Clustering (minPts=4)
clustering_hdbscan(points, neighborhoods, minPts = 4)
```



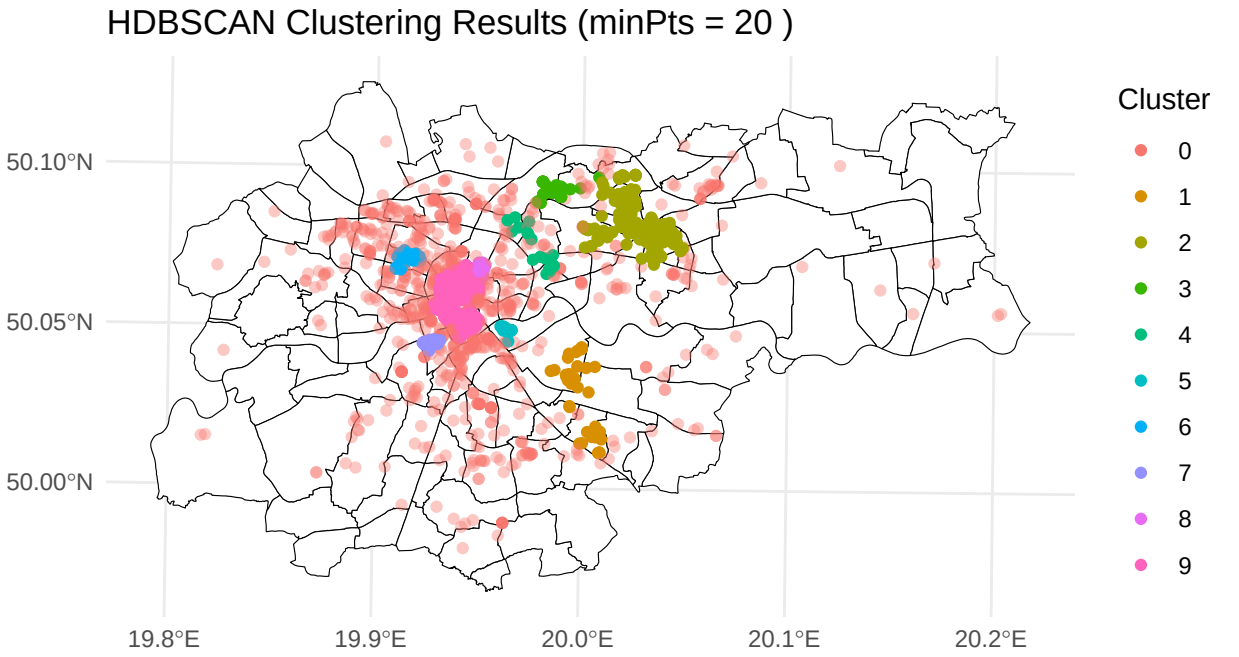
In this example, 24 clusters were formed. It is evident that a large portion of the points were classified into a single cluster.

```r
# HDBSCAN Clustering (minPts=10)
clustering_hdbscan(points, neighborhoods, minPts = 10)
```



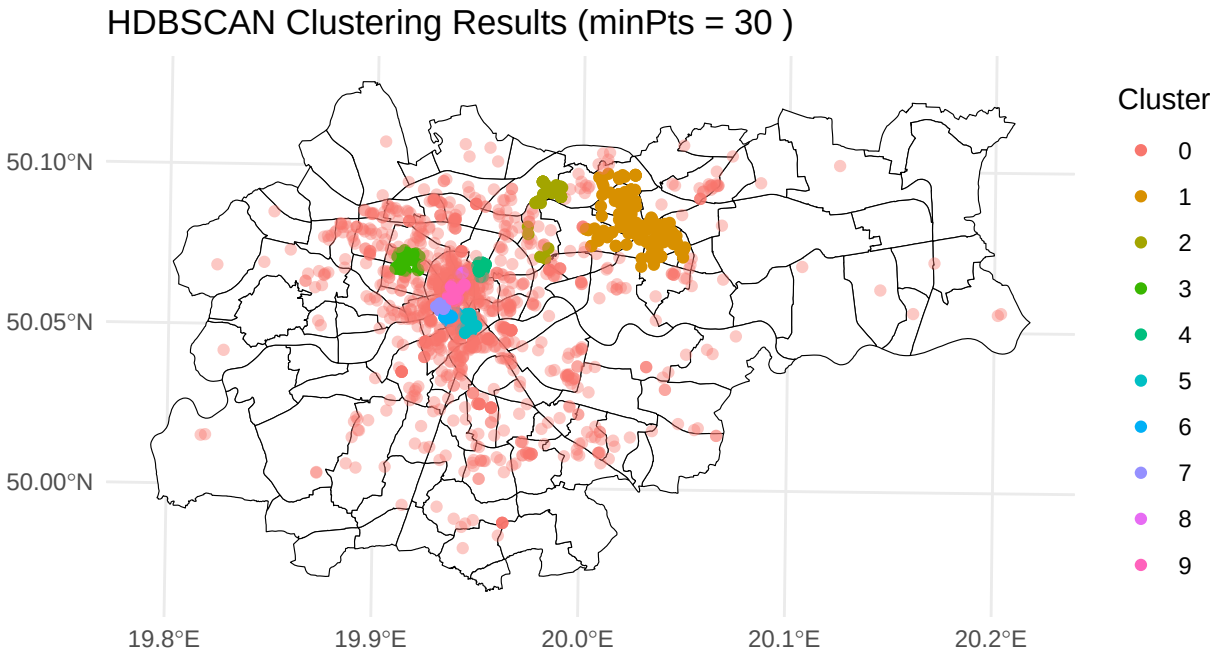
In the next case, more clusters were formed, and the data distribution on the map shows that the clusters are more "organized" (it is easier to identify where the largest concentrations are).

```
# HDBSCAN Clustering (minPts=20)
clustering_hdbscan(points, neighborhoods, minPts = 20)
```



In this case, 9 clusters were formed. The clustering is more restrictive. Only the densest point clusters are visible, while smaller ones were not considered.

```
# HDBSCAN Clustering (minPts=30)
clustering_hdbscan(points, neighborhoods, minPts = 30)
```



In the last example, the vast majority of points were treated as noise due to the high minPts value.

Comparison and Conclusions

Kraków districts with the highest cluster density: The highest intensity of recorded offenses was observed in Stare Miasto, Kleparz, Nowy Świat, Stradom, Piasek Północ, and Piasek Południe.
Additionally, increased intensity was also recorded in the areas of Bieńczyce Nowe, Osiedle Albertyńskie, Prokocim, Nowa Wieś Południe, Rakowice, and Grzegórzki.

In this project, two clustering methods for point data were used: DBSCAN and HDBSCAN. DBSCAN enabled the identification of areas with a high density of clustered points (such as Stare Miasto and Kazimierz). With a larger eps value (e.g., 500), fewer but larger clusters were formed. In contrast, HDBSCAN, with low minPts values (e.g., minPts = 4), identified many small clusters, including less dense areas. Higher minPts values (e.g., minPts = 20 or 30) resulted in a more restrictive cluster selection, eliminating less intense data concentrations. This method was more effective in finding stable clusters in areas with varying densities.

Both DBSCAN and HDBSCAN identified key areas with the highest number of offenses. However, HDBSCAN proved to be a more flexible method (as it did not require setting an eps value, which in DBSCAN could sometimes be difficult to determine alongside an appropriate minPts value) and performed well in identifying clusters in this spatial layout.