# Ebertowski_Maurycy_1_projekt1

MaurycyEbertowski

2024-11-23

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
library(dbscan)
```

```
## Warning: pakiet 'dbscan' został zbudowany w wersji R 4.3.3
```

```
##
## Dołączanie pakietu: 'dbscan'
```

```
## Następujący obiekt został zakryty z 'package:stats':
##
##     as.dendrogram
```

```
library(sf)
```

```
## Warning: pakiet 'sf' został zbudowany w wersji R 4.3.3
```

```
## Linking to GEOS 3.11.2, GDAL 3.8.2, PROJ 9.3.1; sf_use_s2() is TRUE
```

```
punkty <- st_read("C:/Users/1/OneDrive/Pulpit/uni/stare/adp/ebertowski_XYTableToPoint.shp")
```

```
## Reading layer 'ebertowski_XYTableToPoint' from data source
##   'C:\Users\1\OneDrive\Pulpit\uni\stare\adp\ebertowski_XYTableToPoint.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 2000 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 7414345 ymin: 5537814 xmax: 7441750 ymax: 5553321
## Projected CRS: ETRS89 / Poland CS2000 zone 7
```

```
osiedla <- st_read("C:/Users/1/OneDrive/Pulpit/uni/stare/adp/osiedla.shp")
```

```
## Reading layer 'osiedla' from data source
##   'C:\Users\1\OneDrive\Pulpit\uni\stare\adp\osiedla.shp' using driver 'ESRI Shapefile'
## Simple feature collection with 141 features and 30 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 7413437 ymin: 5537344 xmax: 7443955 ymax: 5555031
## Projected CRS: ETRS89 / Poland CS2000 zone 7
```

```
st_crs(punkty)
```

**Sprawdzenie układu współrzędnych**

```
## Coordinate Reference System:
##   User input: ETRS89 / Poland CS2000 zone 7
##   wkt:
## PROJCRS["ETRS89 / Poland CS2000 zone 7",
##     BASEGEOGCRS["ETRS89",
##         DATUM["European Terrestrial Reference System 1989",
##             ELLIPSOID["GRS 1980",6378137,298.257222101,
##                 LENGTHUNIT["metre",1,
##                     ID["EPSG",9001]]]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]]],
##     CONVERSION["unnamed",
##         METHOD["Transverse Mercator",
##             ID["EPSG",9807]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",21,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.999923,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",7500000,
##             LENGTHUNIT["m",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["m",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["m",1]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["m",1]]]
```

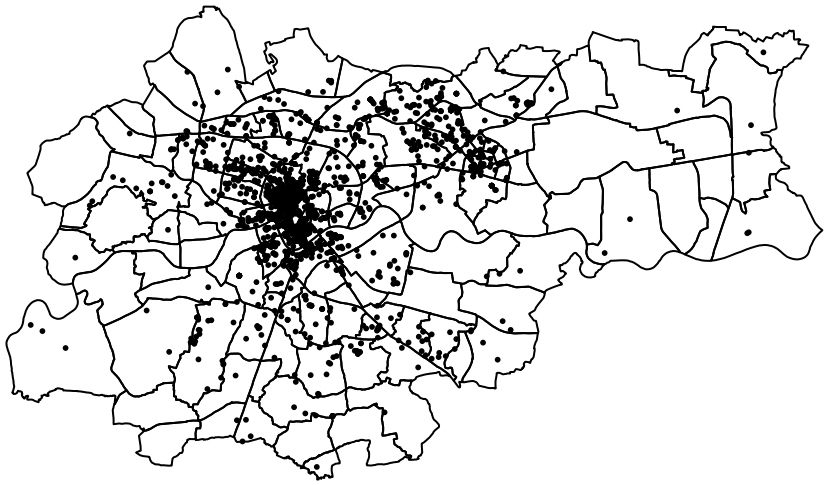
```
st_crs(osiedla)
```

```
## Coordinate Reference System:
##   User input: ETRS89 / Poland CS2000 zone 7
##   wkt:
## BOUNDCRS[
##     SOURCECRS[
##         PROJCRS["ETRS89 / Poland CS2000 zone 7",
##             BASEGEOGCRS["ETRS89",
##                 DATUM["European Terrestrial Reference System 1989",
```

```
##                     ELLIPSOID["GRS 1980",6378137,298.257222101,
##                         LENGTHUNIT["metre",1]]],
##                 PRIMEM["Greenwich",0,
##                     ANGLEUNIT["degree",0.0174532925199433]]],
##             CONVERSION["unnamed",
##                 METHOD["Transverse Mercator",
##                     ID["EPSG",9807]],
##                 PARAMETER["Longitude of natural origin",21,
##                     ANGLEUNIT["degree",0.0174532925199433],
##                     ID["EPSG",8802]],
##                 PARAMETER["Latitude of natural origin",0,
##                     ANGLEUNIT["degree",0.0174532925199433],
##                     ID["EPSG",8801]],
##                 PARAMETER["Scale factor at natural origin",0.999923,
##                     SCALEUNIT["unity",1],
##                     ID["EPSG",8805]],
##                 PARAMETER["False easting",7500000,
##                     LENGTHUNIT["m",1],
##                     ID["EPSG",8806]],
##                 PARAMETER["False northing",0,
##                     LENGTHUNIT["m",1],
##                     ID["EPSG",8807]]],
##             CS[Cartesian,2],
##                 AXIS["easting",east,
##                     ORDER[1],
##                     LENGTHUNIT["m",1]],
##                 AXIS["northing",north,
##                     ORDER[2],
##                     LENGTHUNIT["m",1]],
##             ID["EPSG",2178]]],
##     TARGETCRS[
##         GEOGCRS["WGS 84",
##             DATUM["World Geodetic System 1984",
##                 ELLIPSOID["WGS 84",6378137,298.257223563,
##                     LENGTHUNIT["metre",1]]],
##             PRIMEM["Greenwich",0,
##                 ANGLEUNIT["degree",0.0174532925199433]],
##             CS[ellipsoidal,2],
##                 AXIS["latitude",north,
##                     ORDER[1],
##                     ANGLEUNIT["degree",0.0174532925199433]],
##                 AXIS["longitude",east,
##                     ORDER[2],
##                     ANGLEUNIT["degree",0.0174532925199433]],
##             ID["EPSG",4326]]],
##     ABRIDGEDTRANSFORMATION["Transformation from ETRS89 to WGS84",
##         METHOD["Position Vector transformation (geog2D domain)",
##             ID["EPSG",9606]],
##         PARAMETER["X-axis translation",0,
##             ID["EPSG",8605]],
##         PARAMETER["Y-axis translation",0,
##             ID["EPSG",8606]],
##         PARAMETER["Z-axis translation",0,
##             ID["EPSG",8607]],
```

```
##         PARAMETER["X-axis rotation",0,
##             ID["EPSG",8608]],
##         PARAMETER["Y-axis rotation",0,
##             ID["EPSG",8609]],
##         PARAMETER["Z-axis rotation",0,
##             ID["EPSG",8610]],
##         PARAMETER["Scale difference",1,
##             ID["EPSG",8611]]]]
```

```
plot(osiedla$geometry)
plot(punkty$geometry, pch=16, cex=0.4, add=TRUE)
```



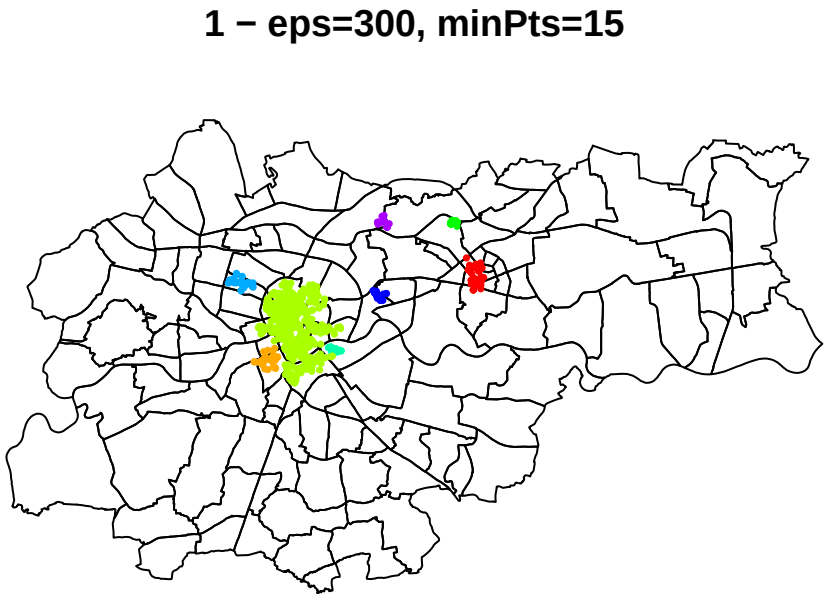
**DBSCAN - Density Based Spatial Clustering of Applications with Noise**

```
punkty_df <- as.data.frame(st_coordinates(punkty))
db <- dbscan(punkty_df, eps=300, minPts=15)
db
```

```
## DBSCAN clustering for 2000 objects.
## Parameters: eps = 300, minPts = 15
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 8 cluster(s) and 780 noise points.
##
```

```
##    0    1    2    3    4    5    6    7    8
##  780   46   43 1023   18   20   30   21   19
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
punkty_df_fltr <- punkty_df[db$cluster != 0,] #filtrujemy szum
plot(osiedla$geometry, main="1 - eps=300, minPts=15")
points(punkty_df_fltr, pch=16, cex=0.5, col=rainbow(length(unique(db$cluster)))[db$cluster[db$cluster !
```




```
#rainbow(n) generuje wektor n równomiernie rozłożonych kolorów, wybieramy klastry != 0
```

```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr, coords = c("X", "Y"), crs = st_crs(osiedla))#zamiana punktów
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)# dopasowanie punktów do osiedli
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

```
##  [1] "Centrum D - Handlowe"          "Ludwin\xf3w"
##  [3] "Stare Miasto"                  "Piasek P\xf3\xb3noc"
##  [5] "Piasek Po\xb3udnie"            "Stare Podg\xf3rze"
##  [7] "Kazimierz"                     "Grzeg\xf3rzki Wsch\xf3d"
```

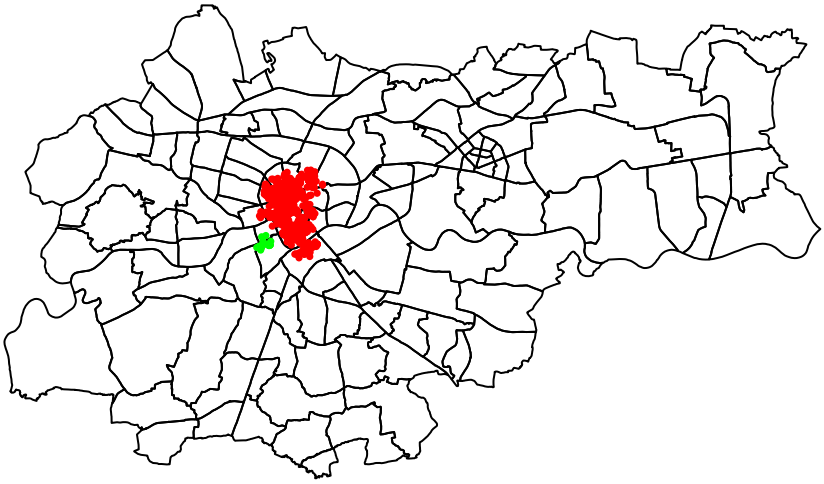
```
##  [9] "Zakrz\xf3wek"                  "Bie\xf1czyce Nowe"
## [11] "Stradom"                       "Grzeg\xf3rzki P\xf3\xb3noc"
## [13] "Grzeg\xf3rzki Zach\xf3d"       "Zab\xb3ocie"
## [15] "Kleparz"                       "P\xf3\xb3wsie Zwierzynieckie"
## [17] "Nowy \x8cwiat"                 "Warszawskie"
## [19] "Centrum C - Zgody"             "Krowodrza - Nowa Wie\x9c"
## [21] "Osiedle Oficerskie"            "Weso\xb3a Zach\xf3d"
## [23] "Czy\xbfyny Park"               "Nowa Wie\x9c Po\xb3udnie"
## [25] "Na Skarpie"                    "Centrum B - Szklane Domy"
## [27] "Urocze"                        "D\xb9bie"
## [29] "Weso\xb3a Wsch\xf3d"           "Pr\xb9dnik Czerwony"
## [31] "Mistrzejowice"                 "Centrum A - Hutnicze Ogrodowe"
## [33] "D\xeabniki"                    "Teatralne"
## [35] "S\xb3oneczne"                  "Krakowiak\xf3w - G\xf3rali"
```

```
db <- dbscan(punkty_df, eps=300, minPts=25)
db
```

```
## DBSCAN clustering for 2000 objects.
## Parameters: eps = 300, minPts = 25
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 2 cluster(s) and 1015 noise points.
##
##    0    1    2
## 1015  958   27
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
punkty_df_fltr <- punkty_df[db$cluster != 0,]
plot(osiedla$geometry, main="2 - eps=300, minPts=25")
points(punkty_df_fltr, pch=16, cex=0.5, col=rainbow(length(unique(db$cluster)))[db$cluster[db$cluster !
```

## 2 − eps=300, minPts=25



```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr, coords = c("X", "Y"), crs = st_crs(osiedla))
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

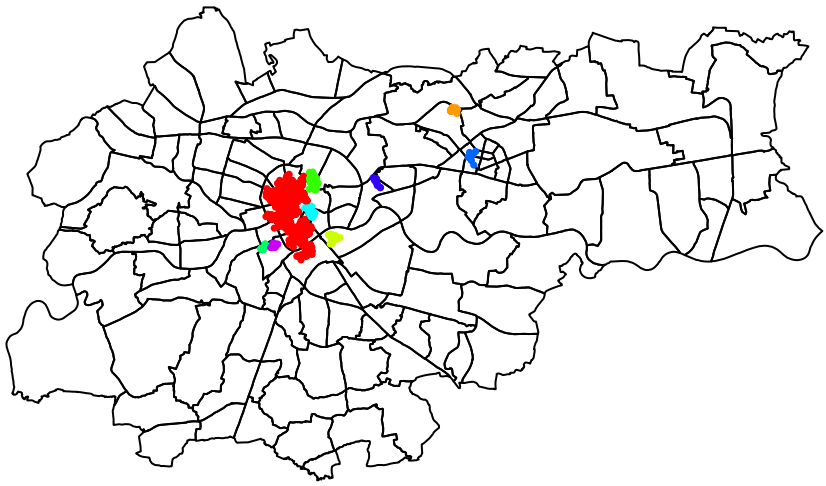
```
##  [1] "Stare Miasto"                 "Piasek P\xf3\xb3noc"
##  [3] "Piasek Po\xb3udnie"           "Stare Podg\xf3rze"
##  [5] "Kazimierz"                    "Zakrz\xf3wek"
##  [7] "Stradom"                      "Grzeg\xf3rzki P\xf3\xb3noc"
##  [9] "Grzeg\xf3rzki Zach\xf3d"      "Kleparz"
## [11] "P\xf3\xb3wsie Zwierzynieckie" "Nowy \x8cwiat"
## [13] "Warszawskie"                  "Ludwin\xf3w"
## [15] "Osiedle Oficerskie"           "Weso\xb3a Zach\xf3d"
## [17] "Weso\xb3a Wsch\xf3d"          "D\xeabniki"
```

```
db <- dbscan(punkty_df, eps=200, minPts=15)
db
```

```
## DBSCAN clustering for 2000 objects.
## Parameters: eps = 200, minPts = 15
```

```
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 9 cluster(s) and 980 noise points.
##
##   0   1   2   3   4   5   6   7   8   9
## 980 833  18  20  64  15  20  20  17  13
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
punkty_df_fltr <- punkty_df[db$cluster != 0,]
plot(osiedla$geometry, main="3 - eps=200, minPts=15")
points(punkty_df_fltr, pch=16, cex=0.5, col=rainbow(length(unique(db$cluster)))[db$cluster[db$cluster !
```




```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr, coords = c("X", "Y"), crs = st_crs(osiedla))
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

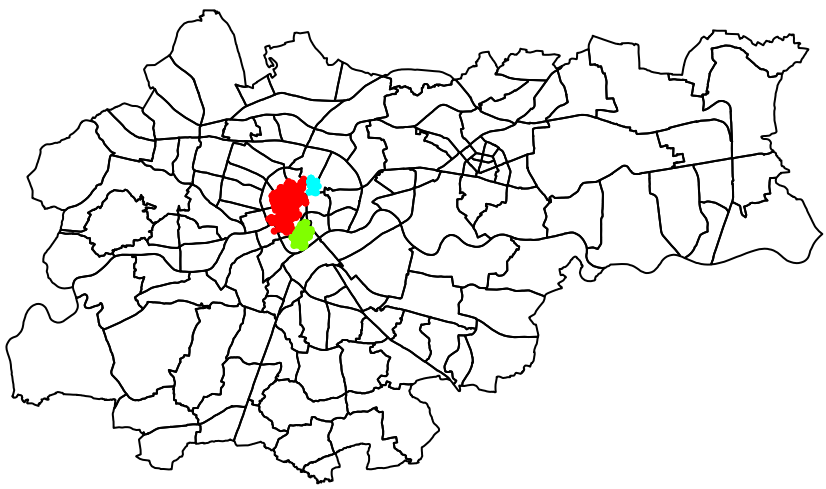
```
##  [1] "Stare Miasto"                "Piasek P\xf3\xb3noc"
##  [3] "Piasek Po\xb3udnie"          "Stare Podg\xf3rze"
##  [5] "Kazimierz"                   "Zakrz\xf3wek"
```

```
##  [7] "Bie\xf1czyce Nowe"            "Stradom"
##  [9] "Grzeg\xf3rzki Zach\xf3d"      "Zab\xb3ocie"
## [11] "Kleparz"                      "Nowy \x8cwiat"
## [13] "Warszawskie"                  "Ludwin\xf3w"
## [15] "Centrum C - Zgody"            "Osiedle Oficerskie"
## [17] "Weso\xb3a Zach\xf3d"          "Urocze"
## [19] "D\xb9bie"                     "P\xf3\xb3wsie Zwierzynieckie"
## [21] "Weso\xb3a Wsch\xf3d"          "Teatralne"
## [23] "D\xeabniki"
```

```
db <- dbscan(punkty_df, eps=200, minPts=25)
db
```

```
## DBSCAN clustering for 2000 objects.
## Parameters: eps = 200, minPts = 25
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 3 cluster(s) and 1182 noise points.
##
##    0    1    2    3
## 1182  619  138   61
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
punkty_df_fltr <- punkty_df[db$cluster != 0,]
plot(osiedla$geometry, main="4 - eps=200, minPts=25")
points(punkty_df_fltr, pch=16, cex=0.5, col=rainbow(length(unique(db$cluster)))[db$cluster[db$cluster !=
```



```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr, coords = c("X", "Y"), crs = st_crs(osiedla))
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

```
##  [1] "Stare Miasto"        "Piasek P\xf3\xb3noc" "Piasek Po\xb3udnie"
##  [4] "Kazimierz"           "Stradom"             "Kleparz"
##  [7] "Nowy \x8cwiat"       "Warszawskie"         "Osiedle Oficerskie"
## [10] "Weso\xb3a Zach\xf3d" "D\xeabniki"
```

**Punkt jest klasyfikowany jako core point jeśli w odległości eps jest co najmniej minPts punktów, punkt jest klasyfikowany jako border point jeśli należy do sąsiedztwa core point, ale sam nie spełnia kryteriów, pozostałe punkty są traktowane jako szum/noise. Klaster jest tworzony przez punkt rdzeniowy i wszystkie punkty osiągalne bezpośrednio z niego.**

**Porównując mapy 1 i 2 oraz 3 i 4 można zobaczyć, że większa wartość parametru minPts odfiltruje małe skupiska danych, a z tych większych wybierze najgęstszy obszar co widzimy na wykresach 1 i 2 lub podzieli go na oddzielne pomniejsze obszary co widać na wykresach 3 i 4. Zwiększenie wartości eps pokaże nam mniej zagęszczone obszary, a te duże o wysokiej gęstości połączy w jeden klaster co pokazują wykresy 1 i 3. Dodatkowo dołączy do takiego klastra dalej oddalone punkty co dobrze widać na wykresach 2 i 4.**

**Zalety DBSCAN:**

**-może znaleźć dowolnie ukształtowane klastry**

**-możliwość ustawienia prametrów eps i minPts, jeśli są dobrze zrozumiane i wiadomo czego chcemy się dowiedzieć o analizowanym zjawisku**

**-wydziela szum i jest odporny na wartości odstające**

**Wady DBSCAN:**

**-wymaga dobrego zrozumienia algorytmu, aby odpowiednio dostosować parametry**

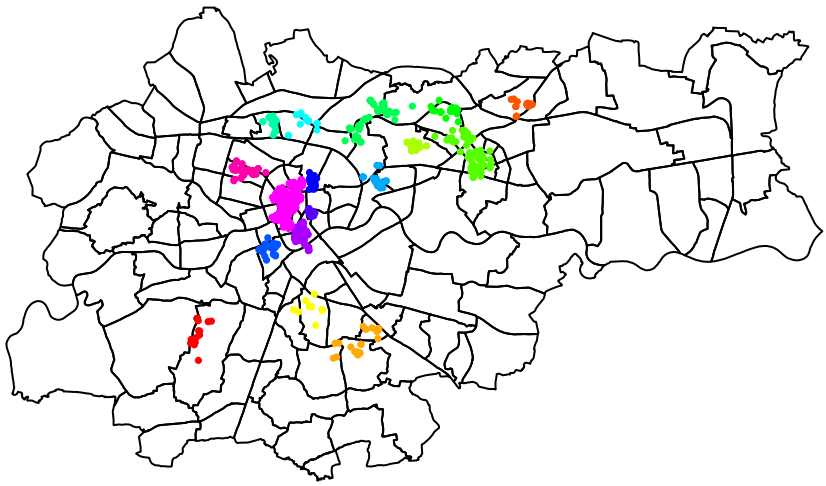
**-punkty graniczne, które są osiągalne z kilku klastrów, będą przydzielone do któregoś z klastrów w zależności od kolejności przetwarzania danych**

**HDBSCAN - Hierarchical Density Based Spatial Clustering of Applications with Noise**

```
hdb <- hdbscan(punkty_df, minPts=15)
hdb
```

```
## HDBSCAN clustering for 2000 objects.
## Parameters: minPts = 15
## The clustering contains 17 cluster(s) and 876 noise points.
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
## 876  16  15  26  16  17  76  28  55  19  20  26  38  60  16 114 547  35
##
## Available fields: cluster, minPts, coredist, cluster_scores,
##                   membership_prob, outlier_scores, hc
```

```
punkty_df_fltr_hdb <- punkty_df[hdb$cluster != 0,]
plot(osiedla$geometry, main="5 - minPts=15")
points(punkty_df_fltr_hdb, pch=16, cex=0.5, col=rainbow(length(unique(hdb$cluster)))[hdb$cluster[hdb$clu
```




```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr_hdb, coords = c("X", "Y"), crs = st_crs(osiedla))
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)
```

```
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

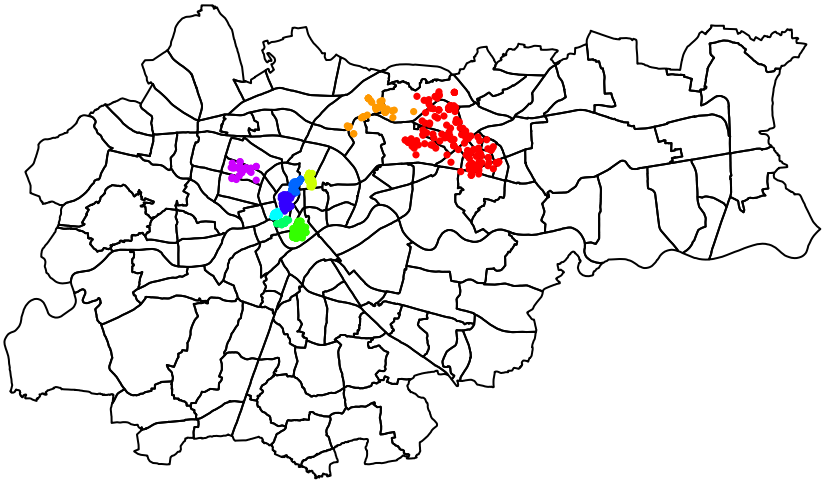
```
##  [1] "Centrum D - Handlowe"            "Ludwin\xf3w"
##  [3] "Stare Miasto"                    "Piasek P\xf3\xb3noc"
##  [5] "Stare Podg\xf3rze"               "Kazimierz"
##  [7] "Zakrz\xf3wek"                    "Bie\xf1czyce Nowe"
##  [9] "Krowodrza Wsch\xf3d"             "Stradom"
## [11] "Czarna Wie\x9c"                  "Grzeg\xf3rzki Zach\xf3d"
## [13] "Kleparz"                         "Nowy \x8cwiat"
## [15] "Krowodrza P\xf3\xb3noc"          "Warszawskie"
## [17] "Rakowice"                        "Kobierzyn"
## [19] "Mistrzejowice"                   "S\xb3oneczne"
## [21] "Pr\xb9dnik Bia\xb3y Po\xb3udnie" "Centrum C - Zgody"
## [23] "Wzg\xf3rza Krzes\xb3awickie"     "Krowodrza - Nowa Wie\x9c"
## [25] "Osiedle Oficerskie"              "\xa3agiewniki"
## [27] "Czy\xbfyny Park"                 "Nowa Wie\x9c Po\xb3udnie"
## [29] "Bie\xf1czyce Stare"              "Na Skarpie"
## [31] "Czy\xbfyny Lotnisko"             "Centrum B - Szklane Domy"
## [33] "Pr\xb9dnik Czerwony"             "Urocze"
## [35] "D\xb9bie"                        "Kurdwan\xf3w"
## [37] "Piasek Po\xb3udnie"              "Skotniki"
## [39] "Wola Duchacka Zach\xf3d"         "Koz\xb3\xf3wek"
## [41] "Piaski"                          "Centrum A - Hutnicze Ogrodowe"
## [43] "D\xeabniki"                      "Olsza"
## [45] "Osiedle Alberty\xf1skie"         "Wola Duchacka Wsch\xf3d"
## [47] "Sp\xf3\xb3dzielcze - Kolorowe"   "Teatralne"
## [49] "Sportowe - Zielone"              "Weso\xb3a Wsch\xf3d"
## [51] "Krakowiak\xf3w - G\xf3rali"      "Kopiec Krakusa"
```

```
hdb <- hdbscan(punkty_df, minPts=25)
hdb
```

```
## HDBSCAN clustering for 2000 objects.
## Parameters: minPts = 25
## The clustering contains 9 cluster(s) and 1201 noise points.
##
##    0    1    2    3    4    5    6    7    8    9
## 1201  177   44   57  109   38   36   86  220   32
##
## Available fields: cluster, minPts, coredist, cluster_scores,
##                   membership_prob, outlier_scores, hc
```

```
punkty_df_fltr_hdb <- punkty_df[hdb$cluster != 0,]
plot(osiedla$geometry, main="6 - minPts=25")
points(punkty_df_fltr_hdb, pch=16, cex=0.5, col=rainbow(length(unique(hdb$cluster)))[hdb$cluster[hdb$clu
```

## 6 − minPts=25



```
punkty_fltr_sf <- st_as_sf(punkty_df_fltr_hdb, coords = c("X", "Y"), crs = st_crs(osiedla))
punkty_osiedla <- st_join(punkty_fltr_sf, osiedla)
nazwy_osiedli <- unique(punkty_osiedla$NAZWA_JEDN)
nazwy_osiedli
```

**Osiedla, na których wyznaczono klastry**

```
##  [1] "Centrum D - Handlowe"          "Stare Miasto"
##  [3] "Kazimierz"                     "Bie\xf1czyce Nowe"
##  [5] "Czarna Wie\x9c"                "Kleparz"
##  [7] "Nowy \x8cwiat"                 "Warszawskie"
##  [9] "Mistrzejowice"                 "S\xb3oneczne"
## [11] "Sportowe - Zielone"            "Centrum C - Zgody"
## [13] "Krowodrza - Nowa Wie\x9c"      "Czy\xbfyny Lotnisko"
## [15] "Osiedle Oficerskie"            "Osiedle Alberty\xf1skie"
## [17] "Nowa Wie\x9c Po\xb3udnie"      "Bie\xf1czyce Stare"
## [19] "Rakowice"                      "Na Skarpie"
## [21] "Centrum B - Szklane Domy"      "Urocze"
## [23] "Pr\xb9dnik Czerwony"           "Centrum A - Hutnicze Ogrodowe"
## [25] "Stradom"                       "Stalowe - Willowe - Wandy"
## [27] "Krakowiak\xf3w - G\xf3rali"    "Sp\xf3\xb3dzielcze - Kolorowe"
## [29] "Teatralne"
```

**Algorytm tworzący hierarchię klastrów o różnej gęstości. Zaczyna od tworzenia małych klastrów, a następnie stopniowo zwiększa eps i łączy je w większe grupy szukając wartości parametru, która daje największą stabilność. Stabilność klastra oznacza, że grupa punktów pozostaje w tej samej grupie, przy zmieniających się kryteriach. Ostatecznie algorytm zwraca zestaw klastrów o różnej gęstości, które najlepiej wpasowują się do zestawu danych.**

**Zalety HDBSCAN:**

**-bardziej elastyczny, może wykrywać klastry w danych, które DBSCAN może zignorować lub pomylić z szumem**

**-mniej wrażliwy na wybór parametrów**

**Wady HDBSCAN**

**-wyższa złożoność obliczeniowa**

**Zwiększając parametr minPts, hdbscan podzieli nam duże, gęste skupiska danych na pomniejsze, te rzadsze które znajdują się blisko siebie połączy w jeden obszar, a te które są małe, rzadkie i daleko od innych skupisk zostaną odfiltrowane.**

**Porównując mapy z zastosowanymi algorytmami dbscan i hdbscan widzimy, że ten drugi pokaże nam więcej informacji. Jest on również prostszy w użyciu lecz wymaga podstawowej wiedzy na temat jego działania, aby go dobrze zinterpretować. Jeżeli zależy nam na stałych parametrach dla całej mapy to użyteczny będzie dbscan, lecz trzeba być pewnym co się chce przeanalizować w celu odpowiedniego ustawienia wartości eps i minPts.**