

# Klasteryzacja przestrzenna danych punktowych

Dorota Zub

## Przygotowanie danych

```
# WCZYTANIE PLIKU SHAPEFILE Z OSIEDLAMI NA TERENIE KRAKOWA
library(sf)
shape<-st_read("osiedla.shp")
```

```
Reading layer 'osiedla' from data source 'D:\RStudio\projekt_analiza\osiedla.shp' using driver 'ESRI Shapefile'
Simple feature collection with 141 features and 30 fields
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: 7413437 ymin: 5537344 xmax: 7443955 ymax: 5555031
projected CRS: ETRS89 / Poland CS2000 zone 7
```

```
library(tmap)
tm_shape(shape) +
  tm_polygons() +
  tm_layout(main.title="Osiedla na terenie Krakowa",
            main.title.position = "center") +
  tm_scale_bar()
```

Osiedla na terenie Krakowa



```
# KOD UKŁADU WSPÓŁRZĘDNYCH
st_crs(shape)$epsg
```

[1] 2178

```
# WCZYTANIE PLIKU Z ZAREJESTROWANYMI WYKROCZENAMI NA TERENIE KRAKOWA
library(readxl)
points <- read_excel("zestaw9.xlsx")

head(points)
```

# A tibble: 6 x 2

	Long	Lat
	<dbl>	<dbl>
1	20.0	50.0
2	19.9	50.1
3	19.8	50.0
4	19.9	50.1
5	20.0	50.0
6	19.9	50.1

```
nrow(points)
```

[1] 2000

```
summary(points)
```

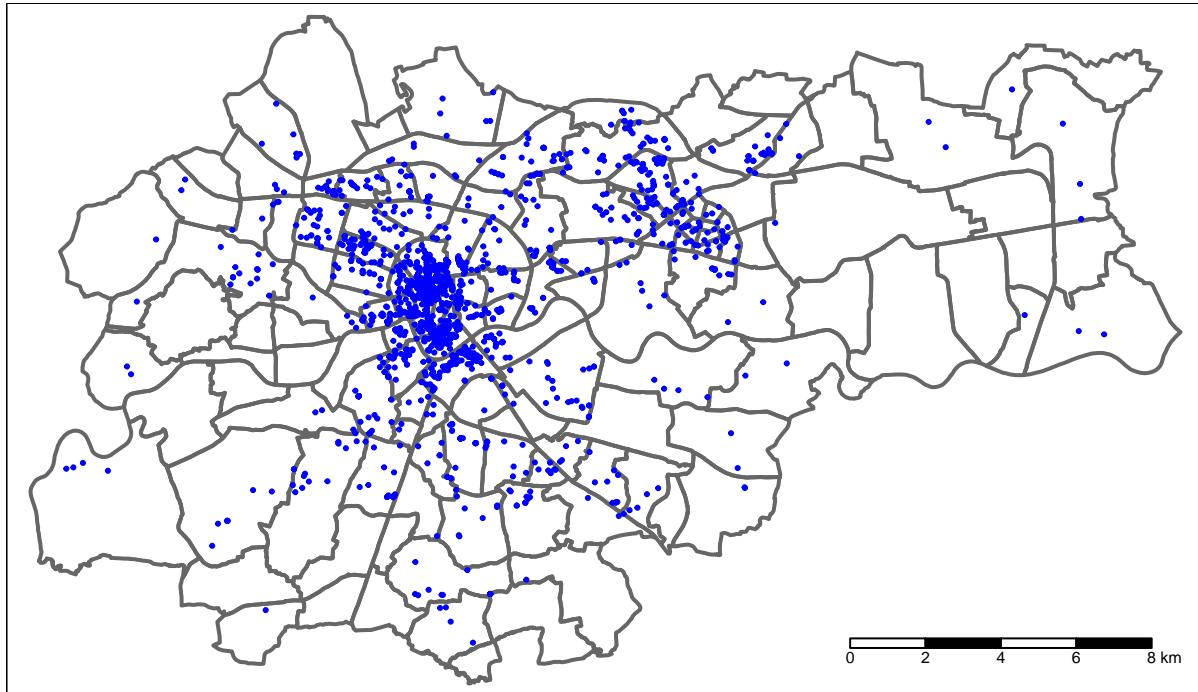
	Long	Lat
Min.	:19.81	:49.98
1st Qu.	:19.93	:50.05
Median	:19.94	:50.06
Mean	:19.95	:50.06
3rd Qu.	:19.96	:50.07
Max.	:20.19	:50.11

```
# STWORZENIE OBIEKTU TYPU SF
points_sf<-st_as_sf(points,coords=c("Long","Lat"), crs="+proj=longlat")

# ZMIANA NA ODPOMIEDNI UKŁAD WSPÓŁRZĘDNYCH
points_sf<-st_transform(points_sf,crs=st_crs(shape))

tm_shape(shape) +
  tm_borders(lwd = 2) +
  tm_shape(points_sf) +
  tm_dots(col="blue") +
  tm_layout(main.title="Zarejestrowane wykroczenia na terenie Krakowa",
            main.title.position = "center") +
  tm_scale_bar()
```

# Zarejestrowane wykroczenia na terenie Krakowa



```
# ZAPISANIE PUNKTÓW DO PLIKU SHAPEFILE  
st_write(points_sf, "D:/RStudio/projekt_analiza/points.shp")
```

Writing layer 'points' to data source 'D:/RStudio/projekt\_analiza/points.shp' using driver 'ESRI Shapefile'  
Writing 2000 features with 0 fields and geometry type Point.

```
# WCZYTANIE PUNKTÓW  
points<-st_read("points.shp")
```

Reading layer 'points' from data source 'D:/RStudio/projekt\_analiza/points.shp' using driver 'ESRI Shapefile'  
Simple feature collection with 2000 features and 1 field  
geometry type: POINT  
dimension: XY  
bbox: xmin: 7414388 ymin: 5538426 xmax: 7441926 ymax: 5553105  
projected CRS: ETRS89 / Poland CS2000 zone 7

```
# DODATKOWO POBRANA MAPA DZIELNIC KRAKOWA ZE STRONY gis-support.pl  
districts<-st_read("dzielnice_Krakowa.shp")
```

Reading layer 'dzielnice\_Krakowa' from data source 'D:/RStudio/projekt\_analiza/dzielnice\_Krakowa.shp' using driver  
Simple feature collection with 18 features and 12 fields  
geometry type: POLYGON  
dimension: XY  
bbox: xmin: 556746.1 ymin: 233811.9 xmax: 587119.6 ymax: 251374.5  
projected CRS: ETRS89\_Poland\_CS92

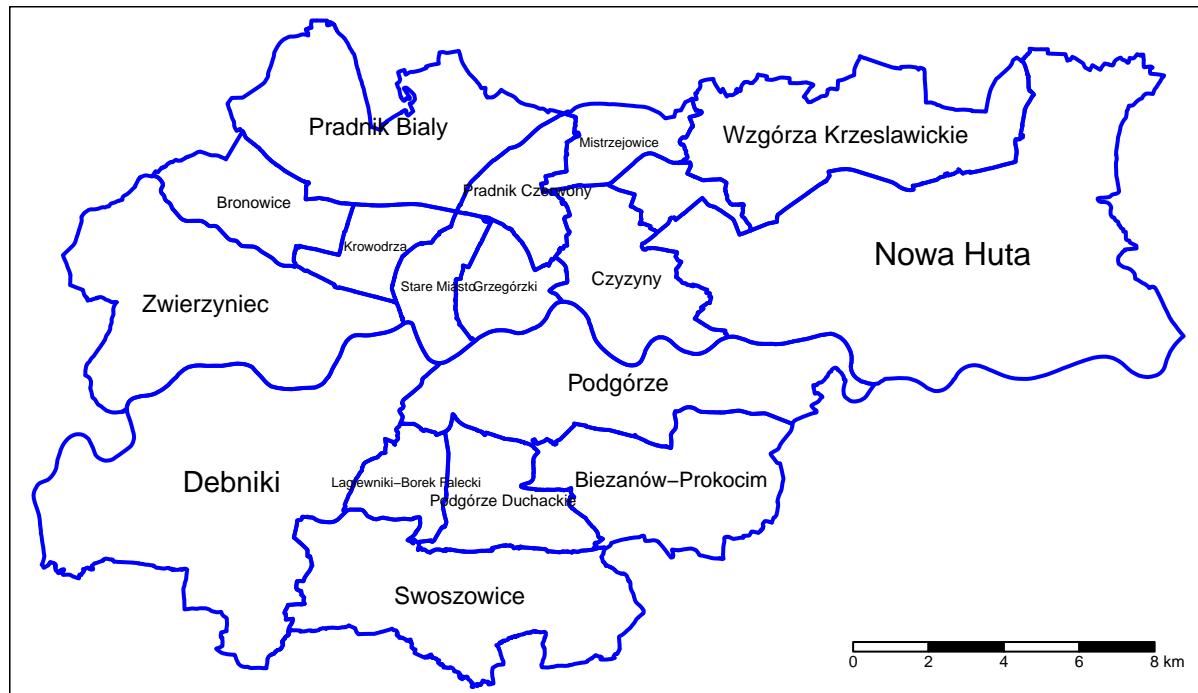
```
districts<-st_transform(districts, crs=st_crs(shape))  
  
shape_map<-tm_shape(shape) +  
  tm_polygons() +
```

```

tm_layout(main.title = "Wykroczenia na terenie Krakowa",
          main.title.position = "center",
          legend.outside = TRUE) +
tm_scale_bar()

tm_shape(districts) +
  tm_borders(lw=2, col = "blue") +
  tm_scale_bar() +
  tm_text("nazwa", size="AREA")

```



## Algorytmy

```

library(dbSCAN)

# PRZYGOTOWANIE MACIERZY KOORDYNATÓW DO WYKONANIA ALGORYTMÓW
points_matrix<-st_coordinates(points)
summary(points_matrix)

```

X	Y
Min. :7414388	Min. :5538426
1st Qu.:7423718	1st Qu.:5546695
Median :7424271	Median :5547759
Mean :7425097	Mean :5547637
3rd Qu.:7425738	3rd Qu.:5548846
Max. :7441926	Max. :5553105

## ALGORYTM DBSCAN

Funkcja `dbSCAN(x, eps, minPTS=5)`, gdzie:

- $x$  jako macierz punktów
- $eps$  jako maksymalny promień sąsiedztwa
- $minPts$  jako minimalna liczba obiektów w regionie  $eps$

**Algorytm DBSCAN** - algorytm gęstościowy grupowania danych, który wyznacza:

- *rdzenie* jako punkty, które w sąsiedztwie o promieniu  $eps$  mają co najmniej  $minPts$  punktów
- *punkty graniczne* jako punkty, które nie są rdzeniami, ale są osiągalne z innego rdzenia
- *szum* jako punkty, które nie należą do żadnej z powyższych grup

## ZALETY DBSACAN

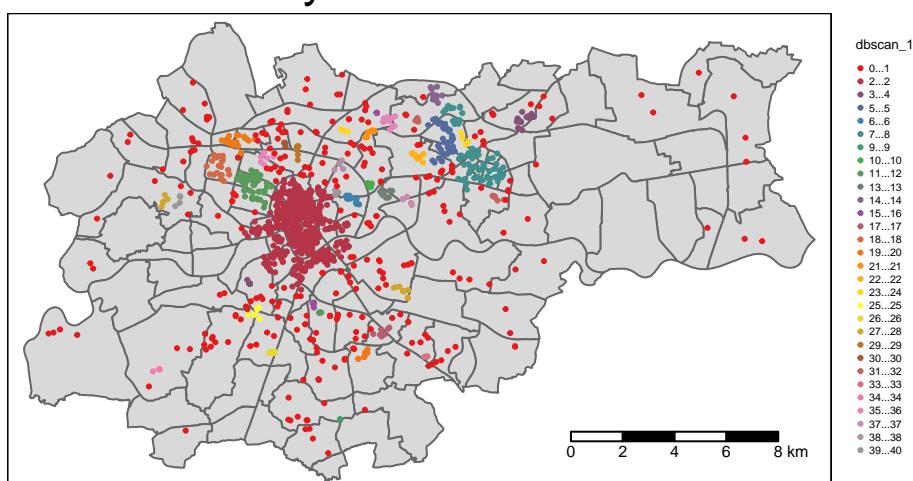
- samodzielnie wyznacza liczbę klastrów
- posiada odporność na obserwacje odstające
- pokazuje szczególnie dobre wyniki dla grup o niewypukłym kształcie

## WADY DBSCAN

- nie grupuje dobrze dla zbiorów z dużymi różnicami w gęstościach
- trudny wybór parametrów

```
dbSCAN_1 <- dbSCAN(points_matrix, eps = 300, minPts = 5)
points$dbSCAN_1 <- as.factor(dbSCAN_1$cluster)
shape_map + tm_shape(points) + tm_dots("dbSCAN_1", palette="Set1")
```

Wykroczenia na terenie Krakowa



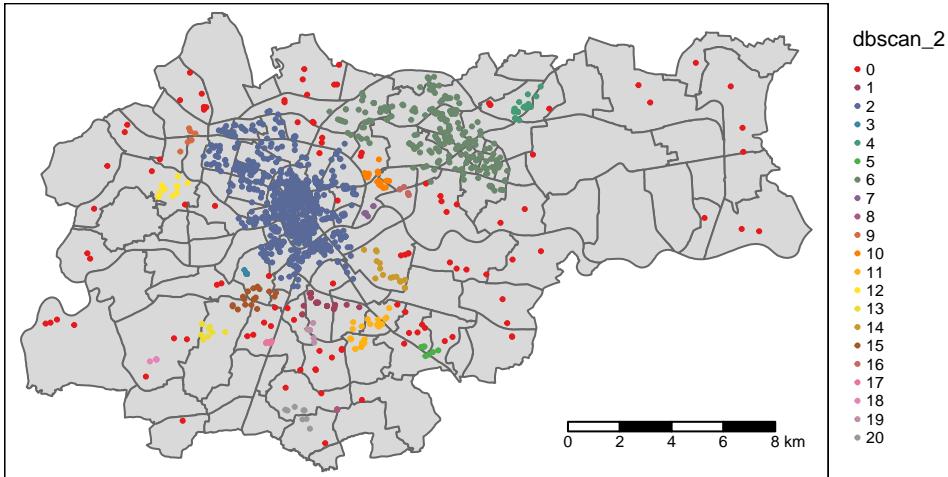
```

# KLASTER Z WARTOŚCIĄ '0' JEST SZUMEM
# WYSTĘPUJE ZBYT DUŻA LICZBA KLASTRÓW, ZWIĘKSZAMY eps

dbSCAN_2 <- dbSCAN(points_matrix, eps = 500, minPts = 5)
points$dbSCAN_2 <- as.factor(dbSCAN_2$cluster)
shape_map + tm_shape(points) + tm_dots("dbSCAN_2", palette="Set1")

```

## Wykroczenia na terenie Krakowa

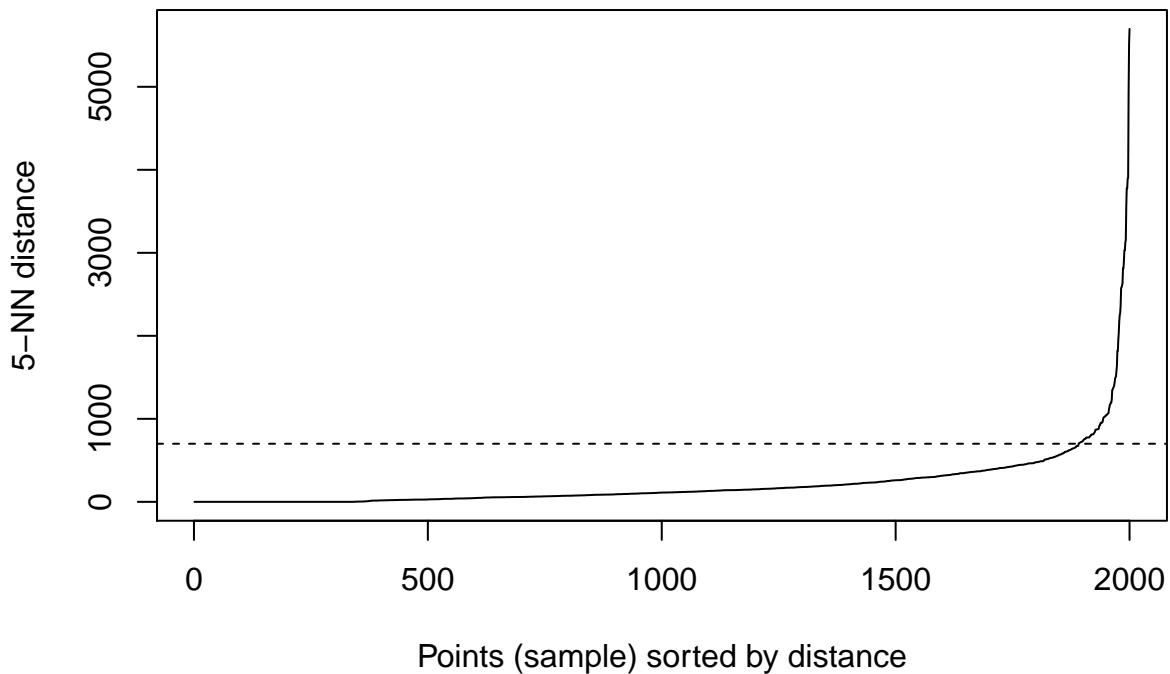


```

# LICZBA KLASTRÓW SIE ZMNIEJSZYŁA

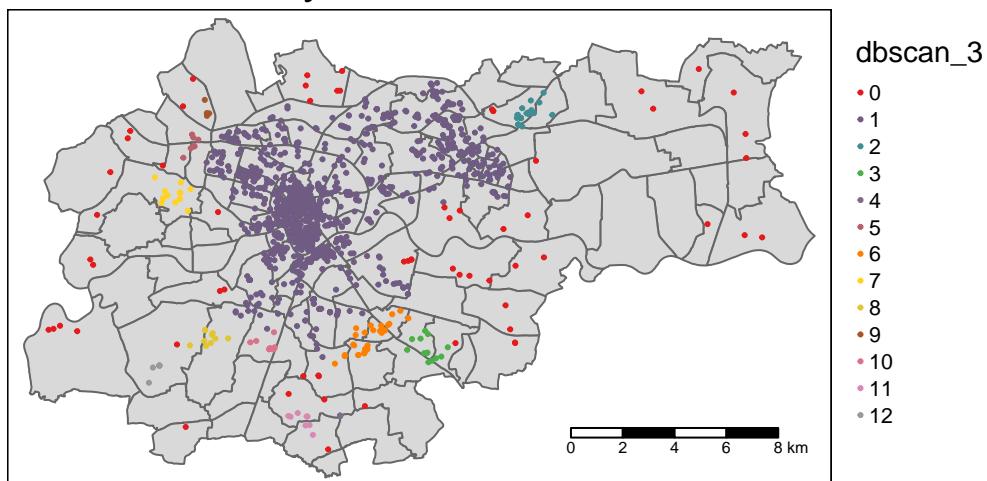
# SZUKAMY OPTYMALNEGO EPS PRZY UŻYCIU FUNKCJI WYZNACZAJĄCEJ ODLEGŁOŚCI DO K-NAJBLIŻSZYCH SĄSIADÓW
kNNdistplot(points_matrix, k = 5)
# WYZNACZENIE "KOLANA" NA WYKRESIE
abline(h = 700, lty = 2)

```



```
dbSCAN_3 <- dbSCAN(points_matrix, eps = 700, minPts = 5)
points$dbSCAN_3 <- as.factor(dbSCAN_3$cluster)
shape_map + tm_shape(points) + tm_dots("dbSCAN_3", palette="Set1")
```

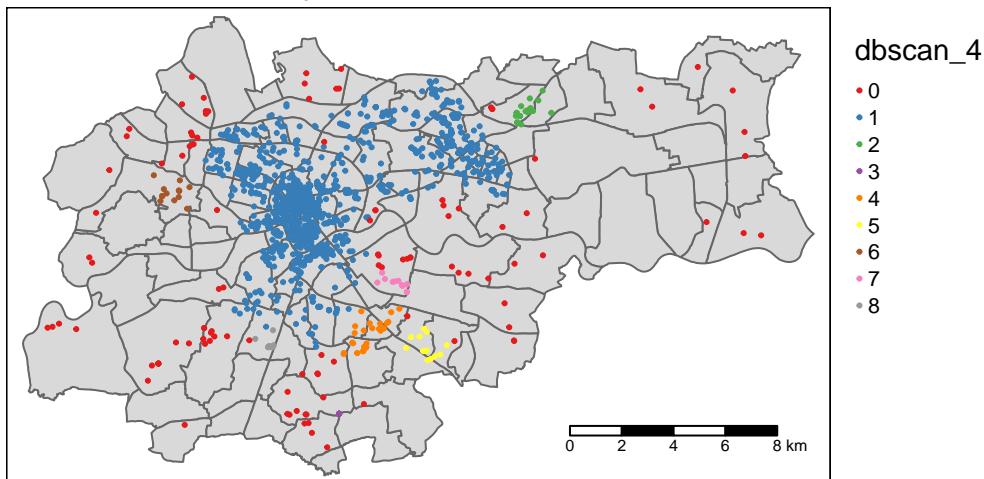
## Wykroczenia na terenie Krakowa



```
# LICZBA KLASTRÓW ZNÓW ULEGŁA ZMNIEJSZENIU, SPRÓBUJMY ZWIĘKSZYĆ minPts
```

```
dbSCAN_4 <- dbSCAN(points_matrix, eps = 700, minPts = 10)
points$dbSCAN_4 <- as.factor(dbSCAN_4$cluster)
shape_map + tm_shape(points) + tm_dots("dbSCAN_4", palette="Set1")
```

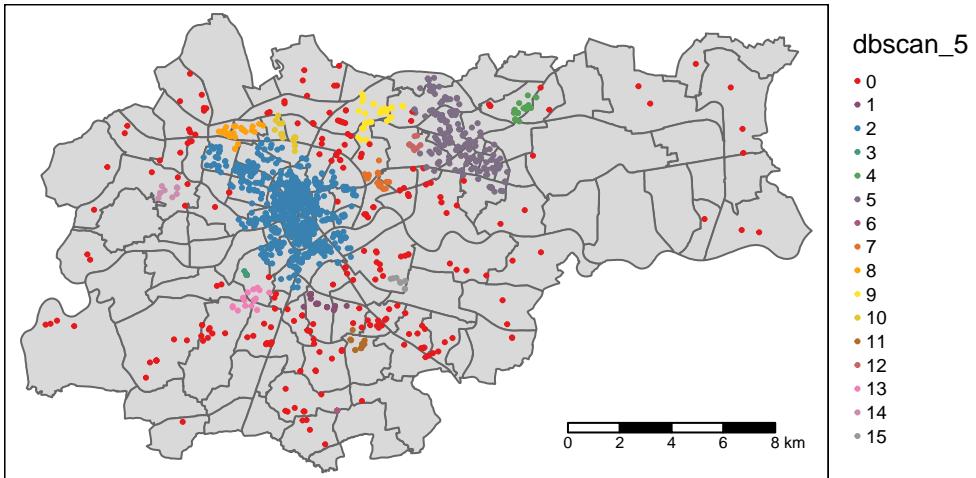
## Wykroczenia na terenie Krakowa



```
# TERAZ ZMNIEJSZMAMY eps PRZY TAKIM SAMYM minPts
```

```
dbSCAN_5 <- dbSCAN(points_matrix, eps = 500, minPts = 10)
points$dbSCAN_5 <- as.factor(dbSCAN_5$cluster)
shape_map + tm_shape(points) + tm_dots("dbSCAN_5", palette="Set1")
```

## Wykroczenia na terenie Krakowa

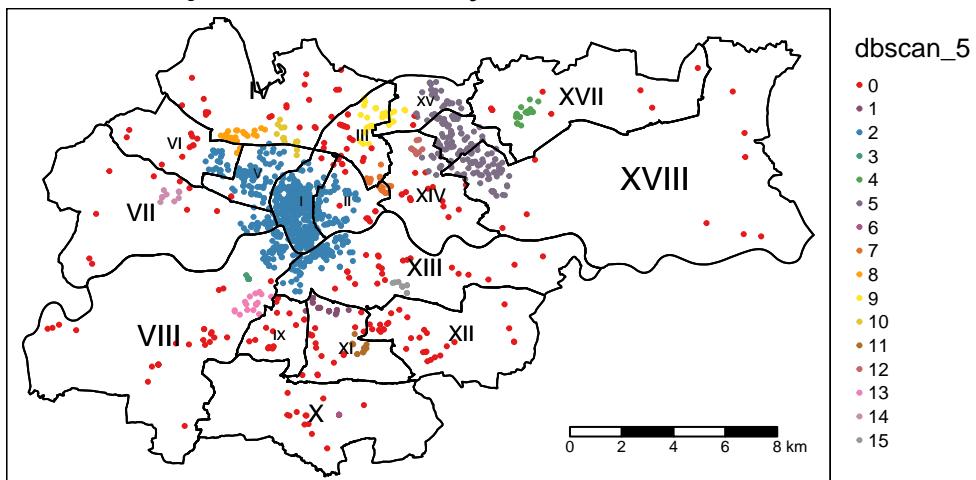


```
# UWAGA, ŻE NAJLEPSZY WYNIK ZOSTAŁ OTRZYMANY PRZY 5 WYNIKU DLA eps = 500 i minPts = 10
dbSCAN_map <- tm_shape(points) + tm_dots("dbscan_5", palette="Set1")

districts_map<-tm_shape(districts) +
  tm_borders(col="black") +
  tm_scale_bar() +
  tm_text("nr_dzielni", size="AREA") +
  tm_layout(main.title="Zarejestrowane wykroczenia na terenie Krakowa",
           main.title.position = "center",
           legend.outside = TRUE)

# DWA RAZY POWTÓRZONE DISTRICT MAP, ABY BYŁ ZASIEGIEM MAPY I OSTATNIĄ WARSTWĄ
districts_map + dbSCAN_map + districts_map
```

## Zarejestrowane wykroczenia na terenie Krakowa



### UTWORZONE KLASTRY:

- Duże: Stare Miasto/Krowodrza i ich okolice oraz na terenie Mistrzejowic/Bieńczyc/Czyżyn
- Małe: Zwierzyniec, Dębniki, Podgórze Duchackie, Wzgórza Krzesławickie, Prądnik Czerwony i Biały

### ALGORYTM OPTICS

Funkcja optics(x, eps, minPTS=5), gdzie:

- $x$  jako macierz punktów
- $eps$  jako maksymalny promień sąsiedztwa
- $minPts$  jako minimalna liczba obiektów w regionie  $eps$

**Algorytm OPTICS** - algorytm gęstościowy grupowania danych, działa podobnie jak dbSCAN, jednak nie przypisuje punktu do danego klastra, lecz przechowuje takie dane dla każdego punktu:

- *odległość rdzenia* - punkty, które w sąsiedztwie o promieniu  $eps$  mają co najmniej  $minPts$  punktów
- *odległość osiągalności* - najmniejsza odległość, gdzie dany punkt jest gęstościowo osiągalny z punktu wewnętrznego

### ZALETY OPTICS

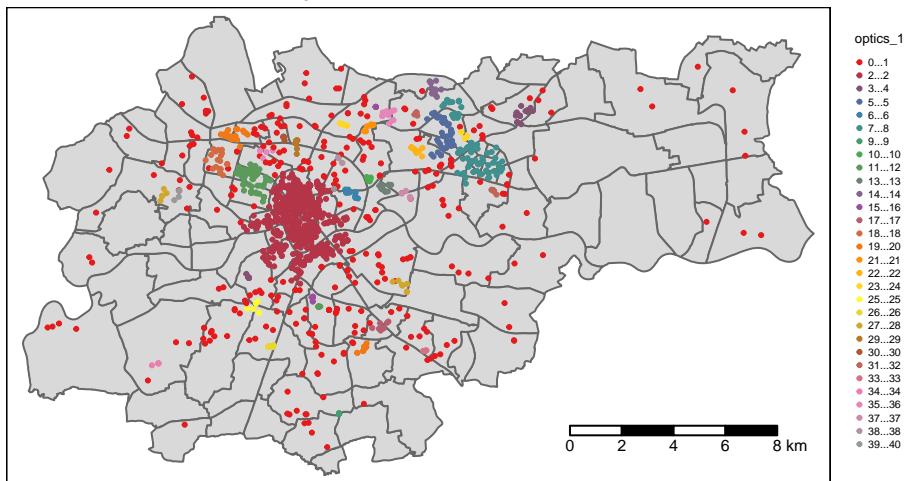
- dobrze grupuje dla zbiorów z dużymi różnicami w gęstościach
- mniejsza wrażliwość na parametry wejściowe

## WADY OPTICS

- trudny wybór parametrów
- brak możliwości obliczenia optymalnego  $\text{eps}$

```
optics_1 <- optics(points_matrix, eps = 300, minPts = 5)
points$optics_1<-as.factor(extractDBSCAN(optics_1, eps_cl = 300)$cluster)
shape_map + tm_shape(points) + tm_dots("optics_1", palette="Set1")
```

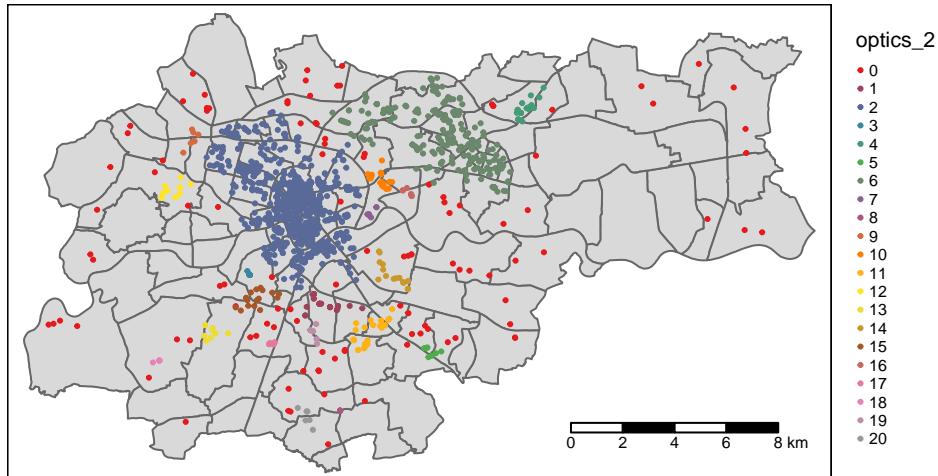
## Wykroczenia na terenie Krakowa



# WYSTĘPUJE ZBYT DUŻA LICZBA KLASTRÓW, ZWIĘKSZAMY  $\text{eps}$

```
optics_2 <- optics(points_matrix, eps = 500, minPts = 5)
points$optics_2<-as.factor(extractDBSCAN(optics_2, eps_cl = 500)$cluster)
shape_map + tm_shape(points) + tm_dots("optics_2", palette="Set1")
```

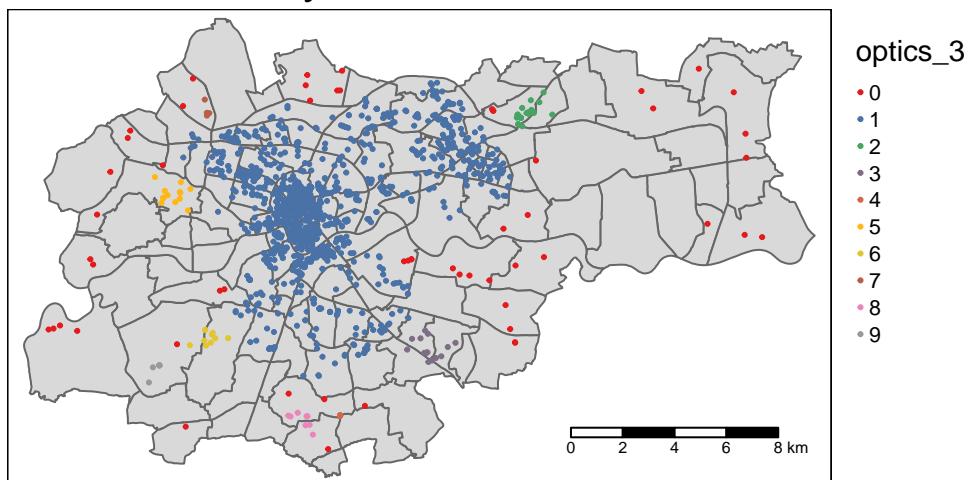
## Wykroczenia na terenie Krakowa



# LICZBA KLASTRÓW SIĘ ZMNIEJSZYŁA

```
optics_3 <- optics(points_matrix, eps = 800, minPts = 5)
points$optics_3<-as.factor(extractDBSCAN(optics_3, eps_cl = 800)$cluster)
shape_map + tm_shape(points) + tm_dots("optics_3", palette="Set1")
```

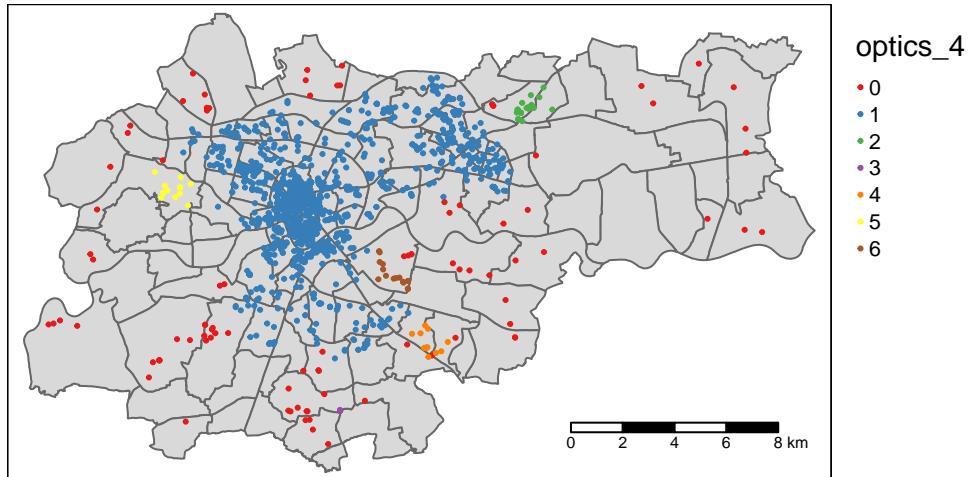
## Wykroczenia na terenie Krakowa



```
# ZBYT MAŁA LICZBA KLASTROW, UTWORZYŁ SIE JEDEN DUŻY KLASTER W CENTRUM
# ZWIĘKSZAMY minPts

optics_4 <- optics(points_matrix, eps = 800, minPts = 10)
points$optics_4<-as.factor(extractDBSCAN(optics_4, eps_cl = 800)$cluster)
shape_map + tm_shape(points) + tm_dots("optics_4", palette="Set1")
```

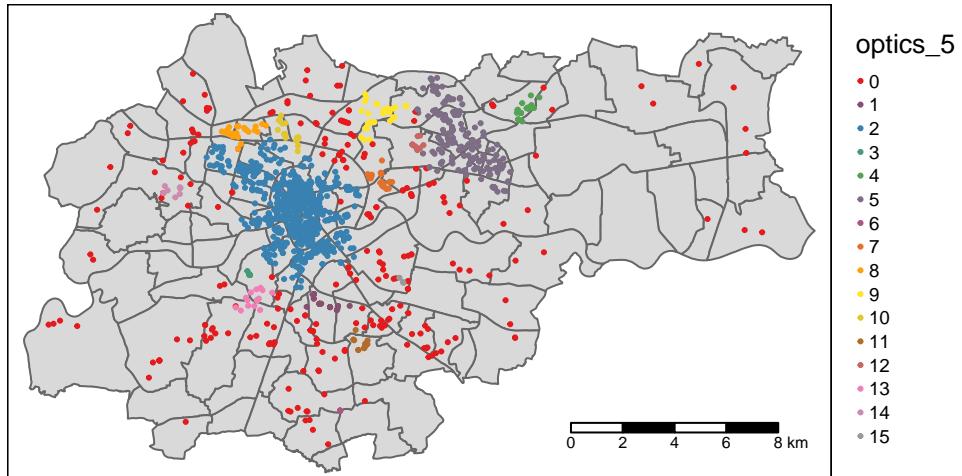
## Wykroczenia na terenie Krakowa



```
# NADAL ZBYT MAŁO KLASTROW, ZMNIEJSZAMY ZATEM eps

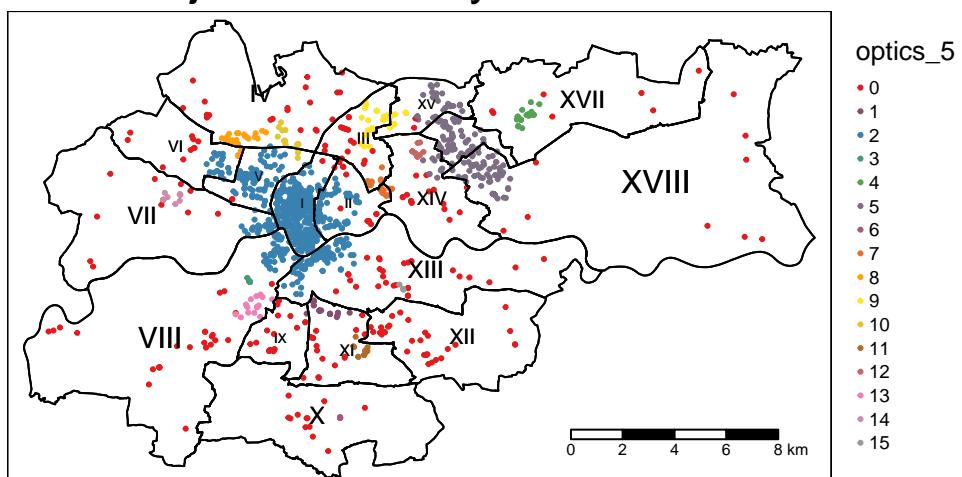
optics_5 <- optics(points_matrix, eps = 500, minPts = 10)
points$optics_5<-as.factor(extractDBSCAN(optics_5, eps_cl = 500)$cluster)
shape_map + tm_shape(points) + tm_dots("optics_5", palette="Set1")
```

## Wykroczenia na terenie Krakowa



```
# UWAGA, ŹE NAJLEPSZY WYNIK ZOSTAŁ OTRZYMANY PRZY 5 WYNIKU DLA eps = 500 i minPts = 10 (TAK JAK PRZY DBSCAN)
optics_map<- tm_shape(points) + tm_dots("optics_5", palette="Set1")
districts_map + optics_map + districts_map
```

## Zarejestrowane wykroczenia na terenie Krakowa



### ALGORYTM HDBSCAN

Funkcja `hdbscan(x, minPTS=5)`, gdzie:

- $x$  jako macierz punktów
- $minPts$  jako minimalna liczba obiektów w regionie  $\epsilon$

**Algorytm HDBSCAN** - algorytm gęstościowy grupowania danych, bardziej rozszerzony od dbscan, tworzy hierarchie, wyznacza:

- *odległość rdzenia* - punkty, które w sąsiedztwie o promieniu  $\epsilon$  mają co najmniej  $minPts$  punktów

### ZALETY HDBSCAN

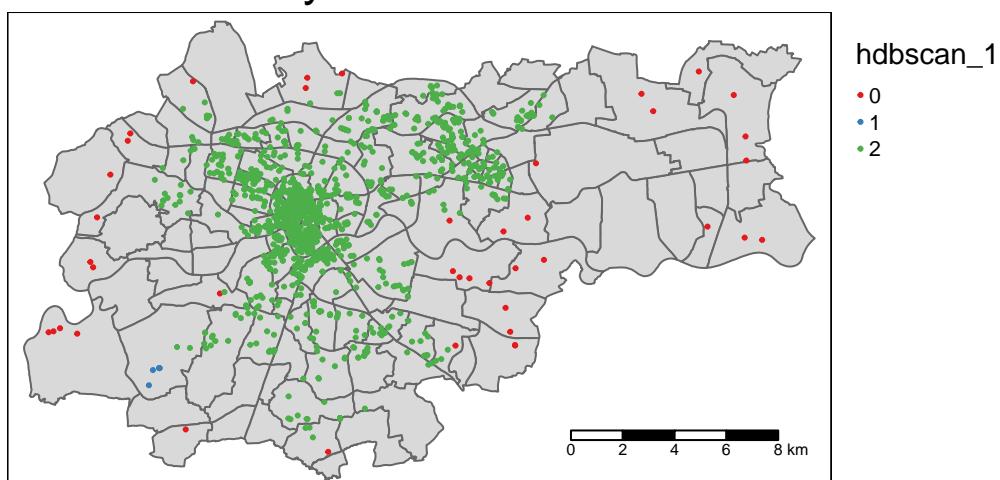
- samodzielnie wyznacza  $\epsilon$
- szybki

### WADY HDBSCAN

- skomplikowany algorytm

```
hdbscan_1 <- hdbscan(points_matrix, minPts = 5)
points$hdbSCAN_1 <- as.factor(hdbscan_1$cluster)
shape_map + tm_shape(points) + tm_dots("hdbscan_1", palette="Set1")
```

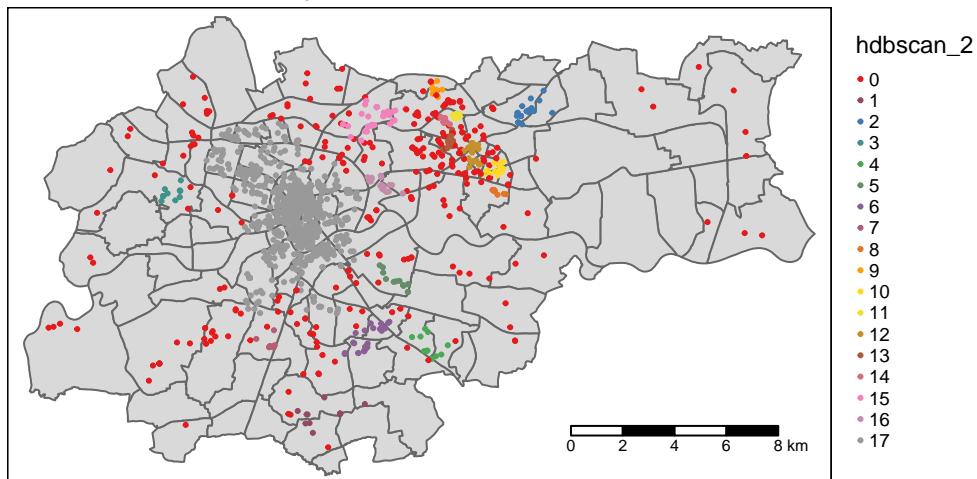
Wykroczenia na terenie Krakowa



```
# UTWORZYŁY SIE TYLKO DWA KLASTRY, SPRÓBUJEMY ZWIĘKSZYĆ minPts
```

```
hdbSCAN_2 <- hdbSCAN(points_matrix, minPts = 10)
points$hdbSCAN_2 <- as.factor(hdbSCAN_2$cluster)
shape_map + tm_shape(points) + tm_dots("hdbSCAN_2", palette="Set1")
```

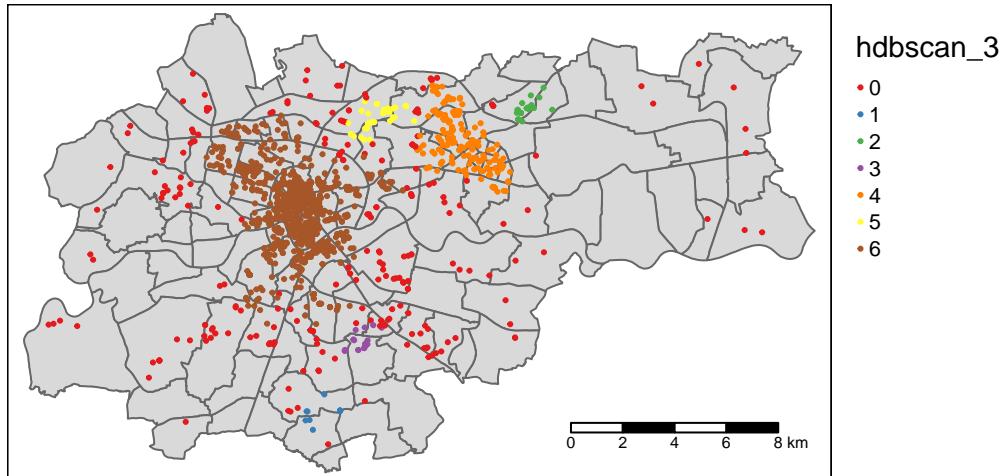
## Wykroczenia na terenie Krakowa



```
# LICZBA KLASTRÓW ZWIĘKSZYŁA SIĘ ZNACZĄCO, SPRÓBUJEMY ZWIĘKSZYĆ JESZCZE RAZ PARAMETR minPts
```

```
hdbSCAN_3 <- hdbSCAN(points_matrix, minPts = 15)
points$hdbSCAN_3 <- as.factor(hdbSCAN_3$cluster)
shape_map + tm_shape(points) + tm_dots("hdbSCAN_3", palette="Set1")
```

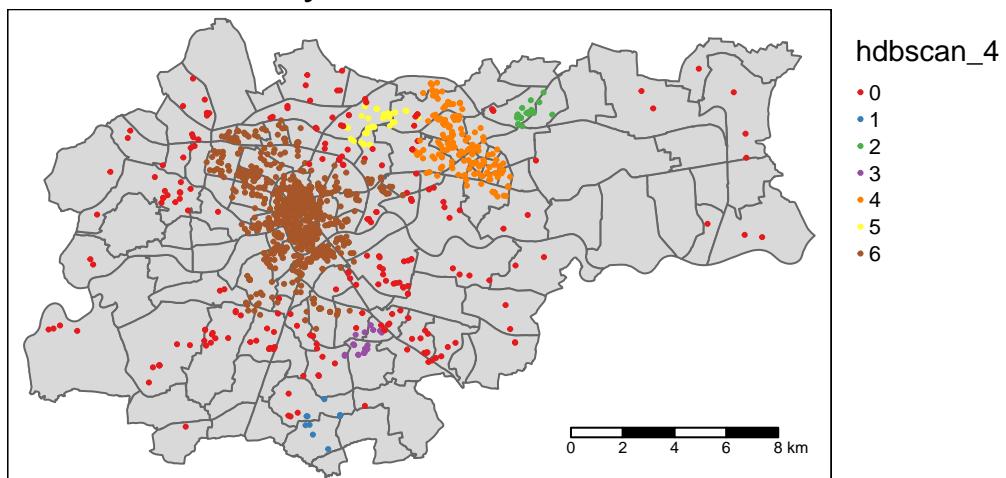
## Wykroczenia na terenie Krakowa



```
# LICZBA KLASTRÓW ULEGŁA ZBYT DUŻEMU ZMNIEJSZENIU, ZATEM ZMNIEJSZAMY minPts
```

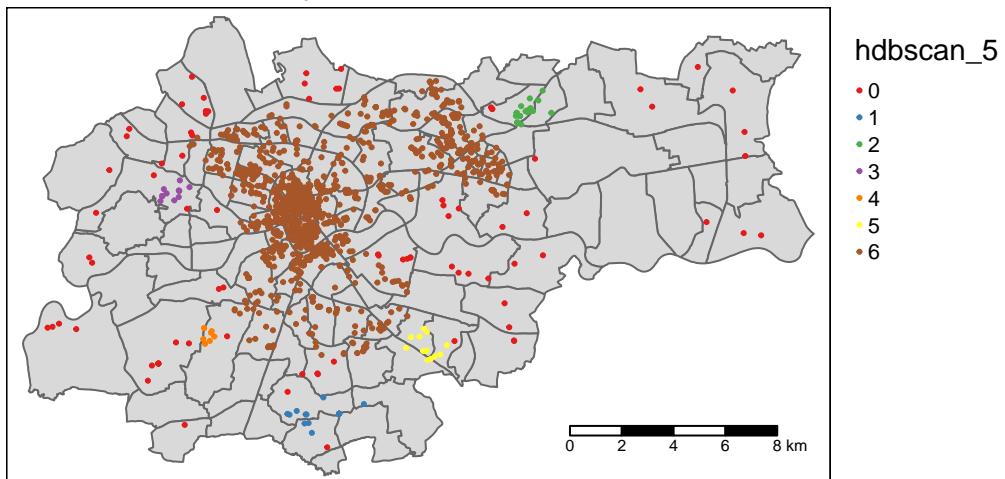
```
hdbSCAN_4 <- hdbSCAN(points_matrix, minPts = 13)
points$hdbSCAN_4 <- as.factor(hdbSCAN_4$cluster)
shape_map + tm_shape(points) + tm_dots("hdbSCAN_4", palette="Set1")
```

## Wykroczenia na terenie Krakowa



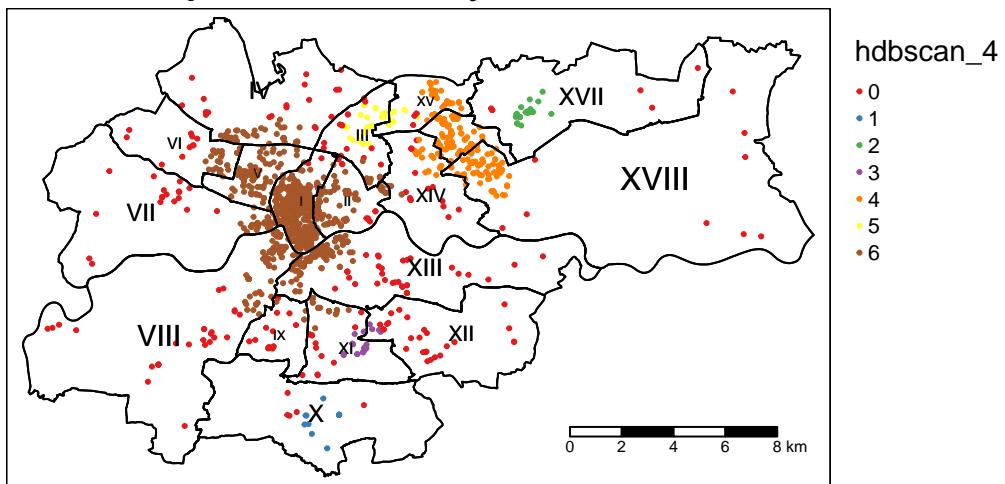
```
# ZNÓW ZMNIEJSZYMY minPts  
  
hdbSCAN_5 <- hdbSCAN(points_matrix, minPts = 8)  
points$hdbSCAN_5 <- as.factor(hdbSCAN_5$cluster)  
shape_map + tm_shape(points) + tm_dots("hdbSCAN_5", palette="Set1")
```

## Wykroczenia na terenie Krakowa



```
# UWAŻAM, ŻE NAJLEPSZY WYNIK ZOSTAŁ OTRZYMANY PRZY 4 WYNIKU DLA minPts = 13  
hdbSCAN_map<-tm_shape(points) + tm_dots("hdbSCAN_4", palette="Set1")  
districts_map + hdbSCAN_map + districts_map
```

## Zarejestrowane wykroczenia na terenie Krakowa



### UTWORZONE KLASTRY:

- Duże: Stare Miasto/Krowodrza i ich okolice oraz na terenie Mistrzejowic/Bieńczyc/Czyżyn
- Małe: Bronowice Małe, Swoszowice, Podgórze Duchackie, Wzgórza Krzesławickie, Prądnik Czerwony

## Podsumowanie

Wszystkie algorytmy dają przybliżone wyniki, jednak algorytm HDBSCAN jest najłatwiejszy w użyciu.