SPRAWOZDANIE

WYKROCZENIA NA OSIEDLACH KRAKOWA

Przygotowała: Kamila Soćko

Podstawowe informacje:

Projekt dotyczy analizy skupień (cluster analysis), czyli eksploracji danych polegającej na podzieleniu zbioru danych na grupy tak, aby elementy, które są do siebie w pewien sposób „podobne" były w tej samej grupie oraz różniły się jak najbardziej od elementów w innych grupach.

Zajmiemy się danymi dotyczącymi wykroczeniami na terenie Krakowa w zależności od osiedli miasta na podstawie algorytmów opartych na gęstościach (density-based algorithms) takich jak DBSCAN, HDBSCAN oraz OPTICS. Te algorytmy nie tylko biorą pod uwagę odległość, ale także gęstość punktów rozłożonych w klastrze.

Do zadania wykorzystuje bibliotekę „dbscan", w której znajdują się funkcje rozwiązujące powyżej wspominane algorytmy oraz dodatkowe funkcje, które umożliwiające jego wykonanie.

Wykorzystane funkcje:

***dbscan** - Density-based spatial clustering of applications with noise:*

dbscan(x, eps, minPts = 5, weights = NULL, borderPoints = TRUE, ...)

Parametry:

- *x – ramka danych*
- *eps – maksymalny promień sąsiedztwa*
- *minPts – minimalna ilość elementów w regionie eps(domyślnie 5 pkt)*
- *borderPoints – jeśli false – punkty graniczne uznawane jako szum*

***hdbscan** – hierarchiczny DBSCAN*

hdbscan(x, eps, minPts)

Parametry:

- *x – ramka danych*
- *minPts – minimalna wielkość klastrów*

***OPTICS** - Ordering points to identify the clustering structure*

optics (x, eps, minPts)

- *x – ramka danych*
- *eps – górna granica promienia sąsiedztwa*
- *minPts – służy do identyfikacji gęstych sąsiedztw*

kNNdist *- Calculate and plot the k-Nearest Neighbor Distance – pozwala na szybkie wyznaczenie odległości k-najbliższych sąsiadów w macierzy punktów. Wykres stworzony na bazie funkcji może być używany do pomóc w znalezieniu odpowiedniej wartości parametru eps dla funkcji DBSCAN (szukanie „kolana” w wykresie).*

PRZYGOTOWANIE DANYCH

Biblioteki:

```r
library(sp)
library(readxl)
library(sf)
library(rgdal)
library(dbscan)
```

ZMIANA WSPÓŁRZĘDNYCH – ZESTAW 4

ZMIANA WSPOLRZEDNYCH MOJEGO ZESTAWU DANYCH PUNKTOWYCH

```r
data <- read_excel("zestaw4.xlsx")
colnames(data) <- c("Lon", "Lat")#zmiana wspolrzednych

coord <- SpatialPoints(cbind(data$Lon, data$Lat), proj4string = CRS("+proj=longlat"))#utworzenie obiektu spatial points potrzebnego do zmiany wspolrzednych dla moich danych
coordUTM <- spTransform(coord, CRS("+init=epsg:2178"))

dataUTM <- data.frame(coordUTM)#zmiana na obiekt data.frame
colnames(dataUTM) <- c("Lon", "Lat")
dataUTM
```

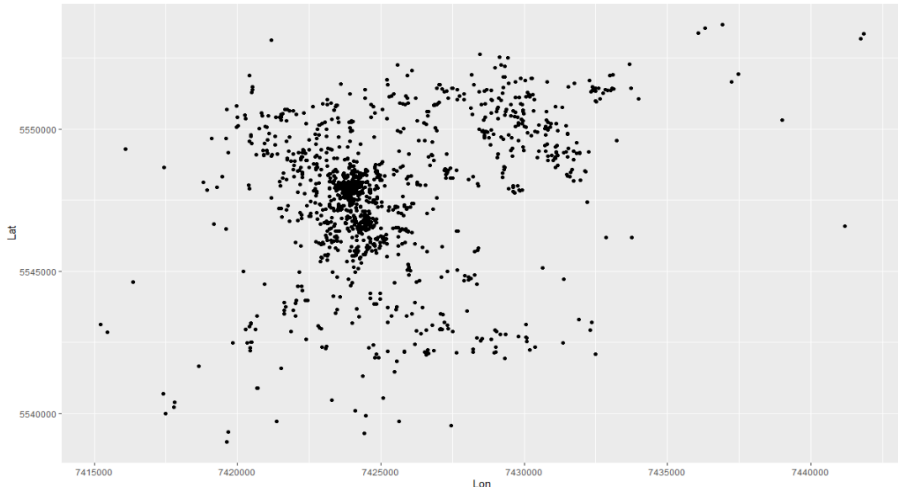
R Console

data.frame
2000 x 2

Lon <dbl>	Lat <dbl>
7423931	5547929
7424385	5548419
7423501	5545742
7422979	5550904
7424845	5547405
7423596	5546033
7424983	5548376

ODCZYT WSPOLRZEDNYCH ZGLOSZONYCH WYKROCZEN

```r
library(ggplot2)
ggplot(dataUTM)+geom_point(aes(Lon,Lat))
```



ZAPIS DO PLIKU

```r
raster::shapefile(coordUTM, "zestaw4.shp", overwrite=T)
```

ODCZYT DANYCH – zestaw4.shp ORAZ osiedla.shp

ODCZYT WSPOLRZEDNYCH ZGLOSZONYCH WYKROCZEN

```r
myData<-readOGR(dsn = ".", layer = "zestaw4")
myData<-data.frame(myData)

library(dplyr)
myData<-select(myData,-optional,-ID)#usuniecie niepotrzebnych kolumn
colnames(myData) <- c("Lon", "Lat")
head(myData)
```

R Console | data.frame 6 x 2

	Lon <dbl>	Lat <dbl>
1	7423931	5547929
2	7424385	5548419
3	7423501	5545742
4	7422979	5550904
5	7424845	5547405
6	7423596	5546033

6 rows

ODCZYT WSPOLRZEDNYCH OSIEDLI

```r
osiedla2<-readOGR(dsn = ".", layer = "osiedla")
os<-data.frame(osiedla2)
os
osiedla2
library(broom)
spdf_fortified <- tidy(osiedla2)#uporzadkowanie danych
spdf_fortified
```

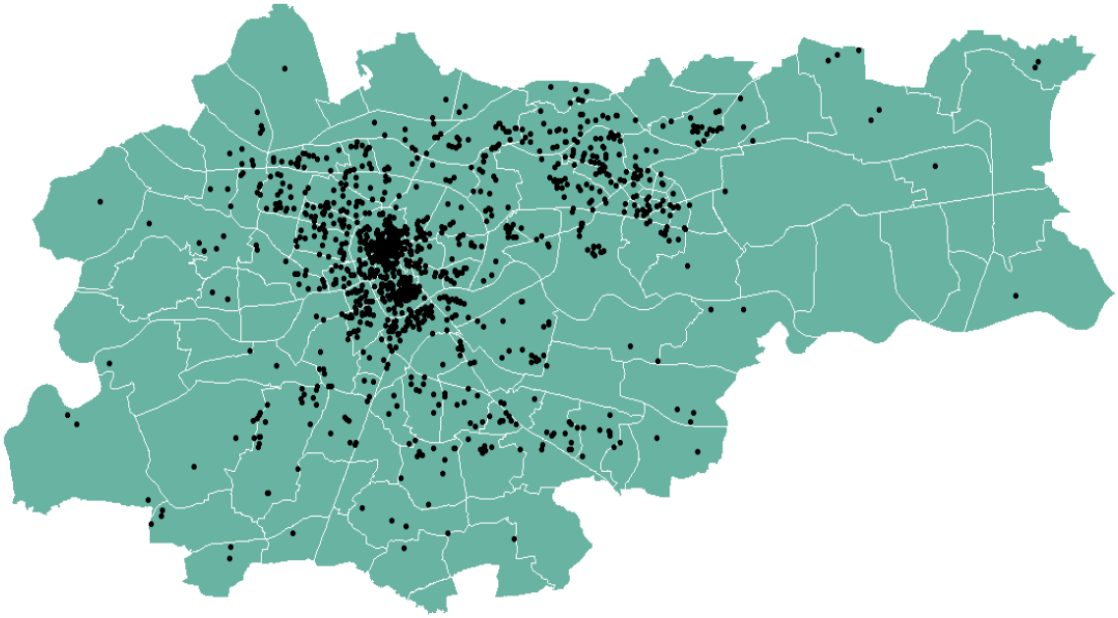
R Console | data.frame 141 x 30 | tbl_df 21218 x 7

long <dbl>	lat <dbl>	order <int>	hole <lgl>	piece <fctr>	group <fctr>	id <chr>
7418583	5552823	1	FALSE	1	0.1	0
7418695	5552977	2	FALSE	1	0.1	0
7418986	5552985	3	FALSE	1	0.1	0
7419022	5553004	4	FALSE	1	0.1	0
7419046	5553024	5	FALSE	1	0.1	0
7419114	5553070	6	FALSE	1	0.1	0
7419402	5553264	7	FALSE	1	0.1	0
7419441	5553285	8	FALSE	1	0.1	0
7419531	5553352	9	FALSE	1	0.1	0
7419588	5553267	10	FALSE	1	0.1	0

1-10 of 21,218 rows | Previous 1 2 3 4 5 6 ... 100 Next

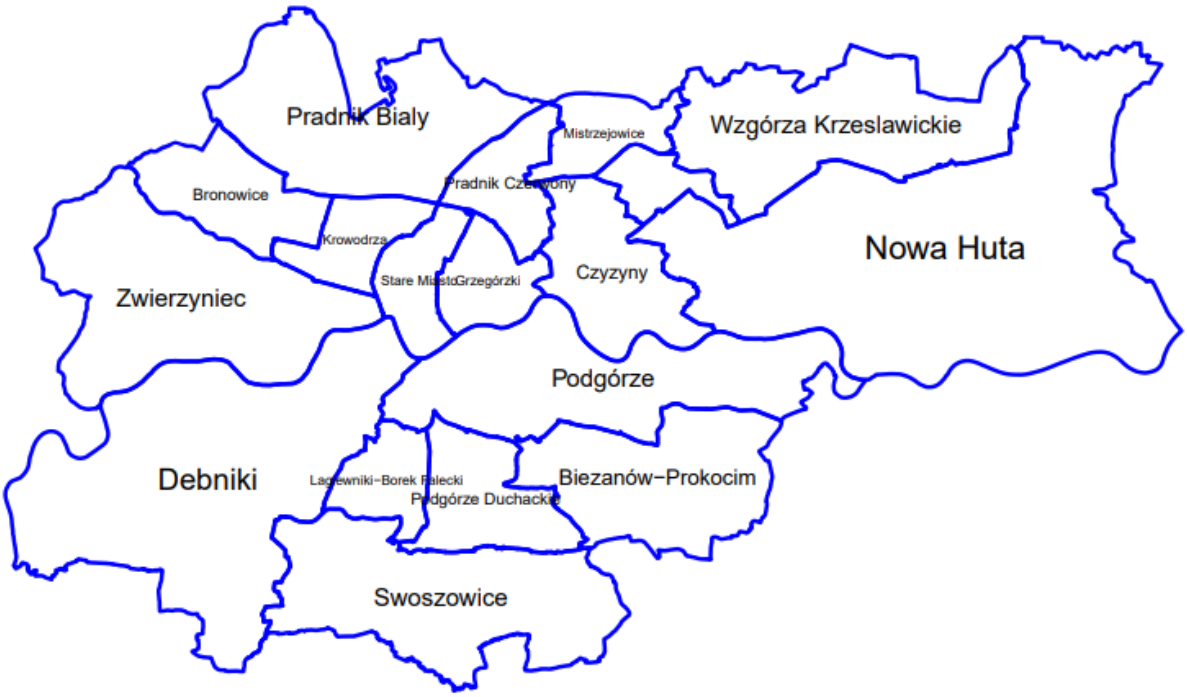
WYKROCZENIA NA OSIEDLACH KRAKOWA PRZEDSTAWIONE NA WYKRESIE

```r
library(ggplot2)
ggplot() +
  geom_polygon(data = spdf_fortified, aes( x = long, y = lat, group = group), fill="#69b3a2", color="white") +
  theme_void() +geom_point(aes(myData$Lon,myData$Lat))
```



NAZWY DZIELNIC

```r
library(tmap)
dzielnice<-readOGR(dsn = ".", layer = "dzielnice_Krakowa")
dzielnice2<- tm_shape(dzielnice) +
tm_borders(lw=2, col = "blue")
dzielnice2+tm_text("nazwa", size="AREA")
```



KLASTERYZACJA

ALGORYTM DBSCAN

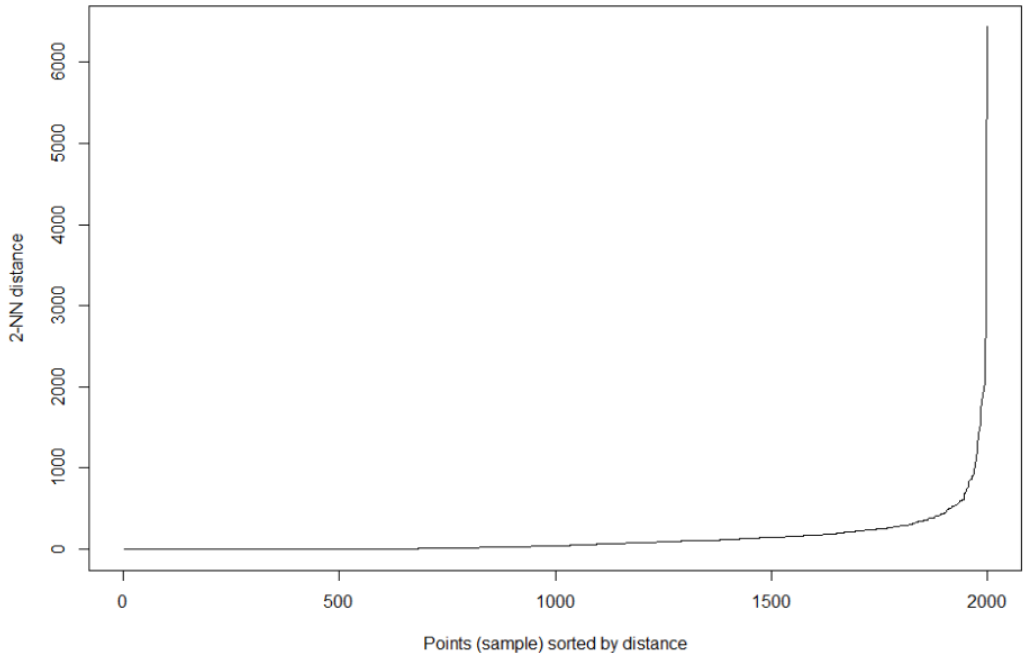
Działanie:

Główną ideą tego algorytmu jest to, że jeśli konkretny punkt należy do klastra to powinien znajdować się w pobliżu wielu innych punktów w tym klastrze. Na początku wybierane są dwa parametry – eps oraz minPts. Następnie wybierany jest dowolny punkt w zbiorze danych. Jeśli w odległości eps od tego punktu znajduje się więcej niż minPts punktów to wszystkie te punkty należą do klastra. Później klaster jest rozszerzany sprawdzając czy wszystkie nowe punkty mają również więcej niż minPts punktów w odległości eps. Jeśli tak jest to klaster jest powiększany. Jeśli zabraknie nam punktów do dodania wybieramy kolejny nowy arbitralny punkt od którego zaczynamy proces od początku. Jeśli wybrany przez nad punkt ma mniej niż minPts punktów w odległości eps to uznawany jest za szum i nie należy do żadnego klastra.

Kod i wyniki

Z wykorzystaniem funkcji kNNdistplot:

```
dimension<-dim(myData)[2]
minPts=dimension+1#3
kNNdistplot(myData, k = minPts-1)
```

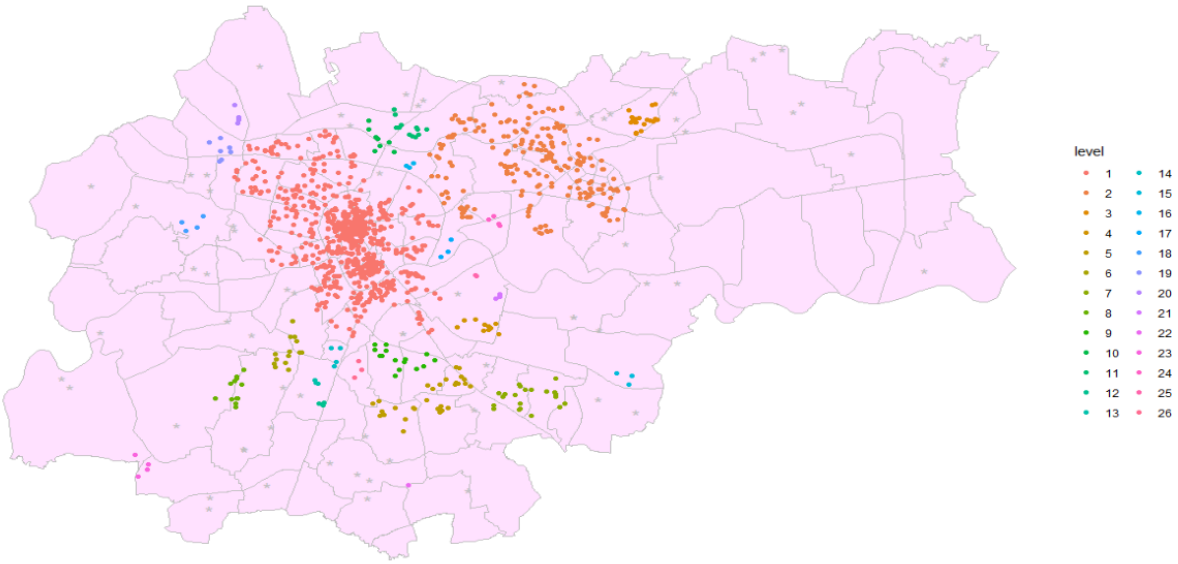


„Kolano" na poziomie około 500, zatem wybieram eps=500, minPts= 3

```
db<-dbscan(myData,eps=500,minPts=3, borderPoints=TRUE)
noise<-myData[db$cluster==0,]#dane okreslone jako szum
data<-myData[db$cluster!=0,]#dane bez szumu
clusters<-as.factor(db$cluster)
level<-clusters[clusters!=0]#wedlug tego grupujemy nie uwzgledniajac danych uznanych jako szum
ggplot() +
  geom_polygon(data = spdf_fortified, aes( x = long, y = lat, group = group), fill="thistle1", color="grey") +
  theme_void() +geom_point(aes(data$Lon,data$Lat, colour = level))+geom_point(aes(noise$Lon,noise$Lat), pch="*", size=5, col="grey")
#szum rysujemy jako szare gwiazdki
```

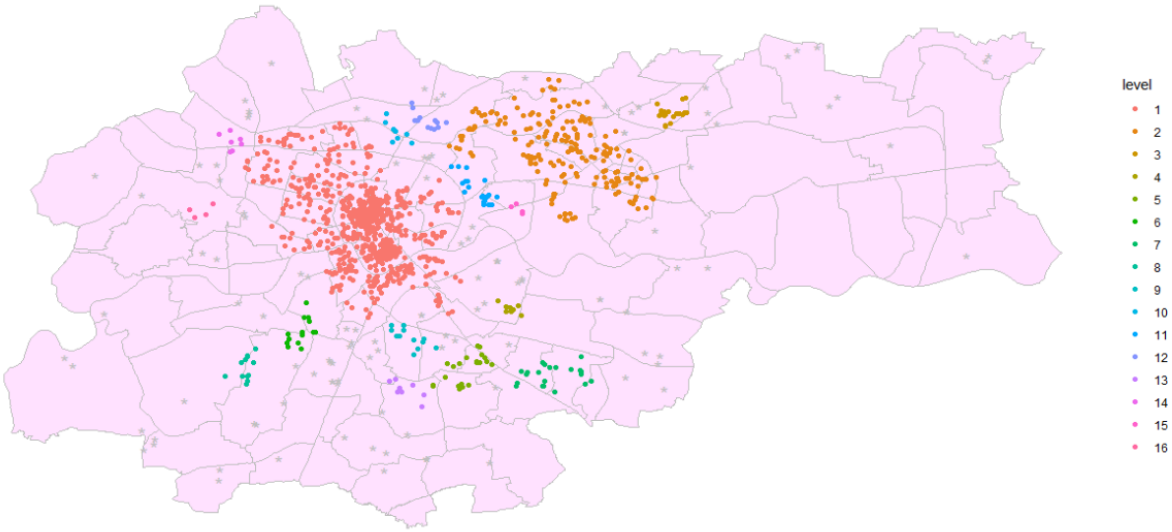
1. *eps=500, minPts=3*

   Jeden duży klaster w centrum i na prawo, zmienię parametry.

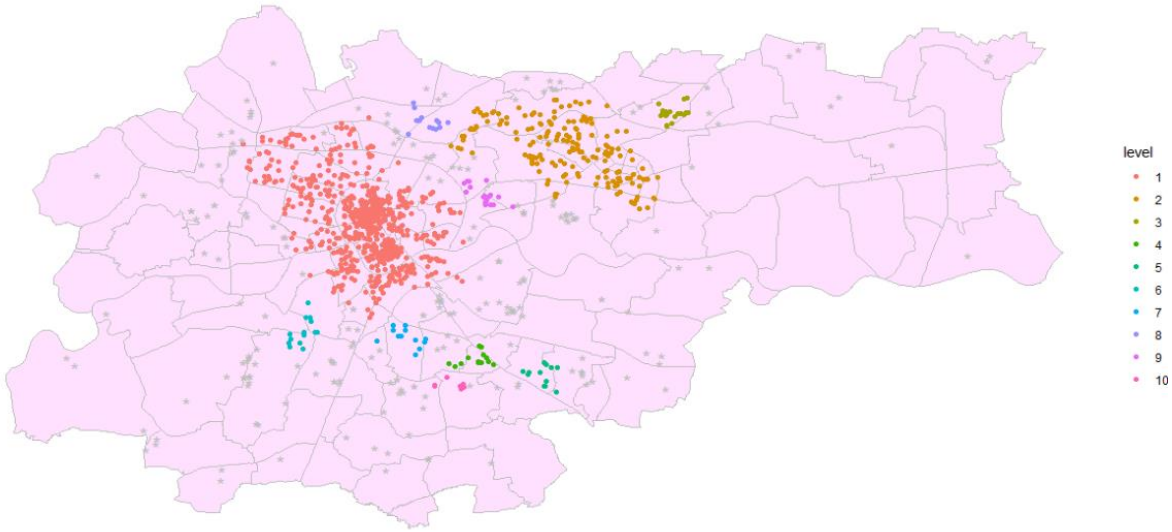


2. *eps=510, minPts=6*

   Lepszy wynik, więcej punktów szumu, ale lepszy podział na klastry.



3. *eps=600, minPts=12*
   Wynik gorszy od poprzedniego, więcej szumu, niektóre grupy nie utworzyły klastrów tylko uznane zostały jako szum.



Porównanie wyników:

Algorytm dla różnych parametrów wydzielił różną ilość klastrów. Najwięcej klastrów z najmniejszym szumem wydzieliło się dla opcji nr 1. Największy szum był dla opcji nr 3, jednakże najbardziej satysfakcjonujący wynik moim zdaniem dała opcja nr 2. W miejscach, w których było mało punktów, ale blisko siebie również klastry zostały utworzone, a nie uznane jako szum.

Dzielnice dla najlepszej opcji(2):

Jeden większy na terenie Nowej Huty, Bieńczyc, Mistrzejowic oraz Czyżyn. Inny na terenie Starego Miasta, Grzegórzki, Krowodrza i część dzielnicy Dębniki. Mniejszy na Wzgórzach Krzesławickich, Prądnik Biały oraz Prądnik Czerwony. Dwa mniejsze również na Dębnikach. Mniejsze również na Podgórzu Duchackim, Bieżanów-Prokocim oraz na Podgórzu.

Wady	*Zalety*
Liczba klastrów zależy od liczby obserwacji oraz dobranych parametrów	*Odporny na wpływ wartości odstających*
Brak stałych i dobrze określonych sposobów na wybór odpowiednich parametrów	*Szybkie działanie*
Nie bierze pod uwagę miar o różnych gęstościach	*Daje możliwość definiowania wielu miar odległości*
Słabo wydajny	*Dobrze radzi sobie w grupowaniu danych o różnym kształcie*

ALGORYTM HDBSCAN

Działanie

Głównym założeniem jest to, że zamiast szukać klastrów o określonym kształcie, szuka regionów danych, które są gęstsze niż otaczająca je przestrzeń. Również poszukuje odległości każdego punktu od jego sąsiadów. Pobierając wiele zestawów poziomów o różnych wartościach gęstości, *otrzymujemy hierarchię.*

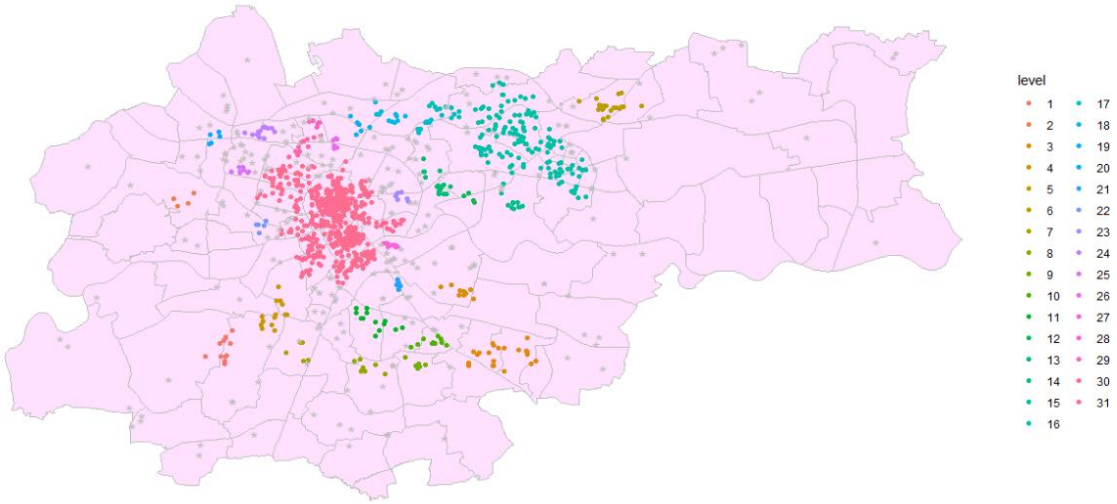
Wymaga podania tylko minimalnego rozmiaru klastra- czyli jak duży musi być klaster, aby mógł zostać utworzony. Określa, które klastry są dla Ciebie ważne na podstawie rozmiaru.

Kod i wyniki

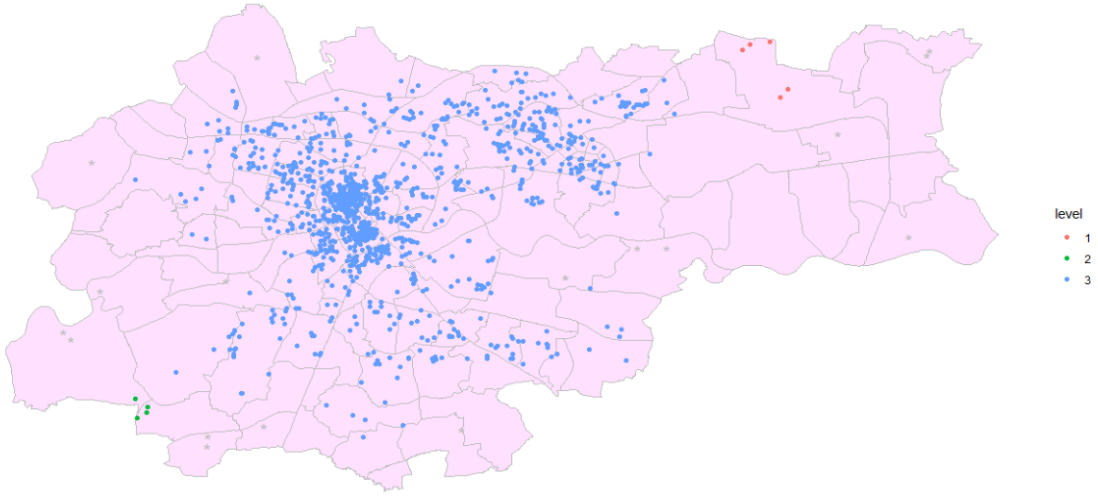
```
db <- hdbscan(myData, minPts =6)

noise<-myData[db$cluster==0,]
data<-myData[db$cluster!=0,]
clusters<-as.factor(db$cluster)
level<-clusters[clusters!=0]
ggplot() +
  geom_polygon(data = spdf_fortified, aes( x = long, y = lat, group = group), fill="thistle1", color="grey") +
  theme_void() +geom_point(aes(data$Lon,data$Lat, colour = level))+geom_point(aes(noise$Lon,noise$Lat), pch="*", size=5, col="grey")
```

1. *minPts=6*
   Podział wydaje się realny, klastry zostały wydzielone na większą ilość, ale widać, że w gęstszych terenach punkty należą do tego samego klastra.

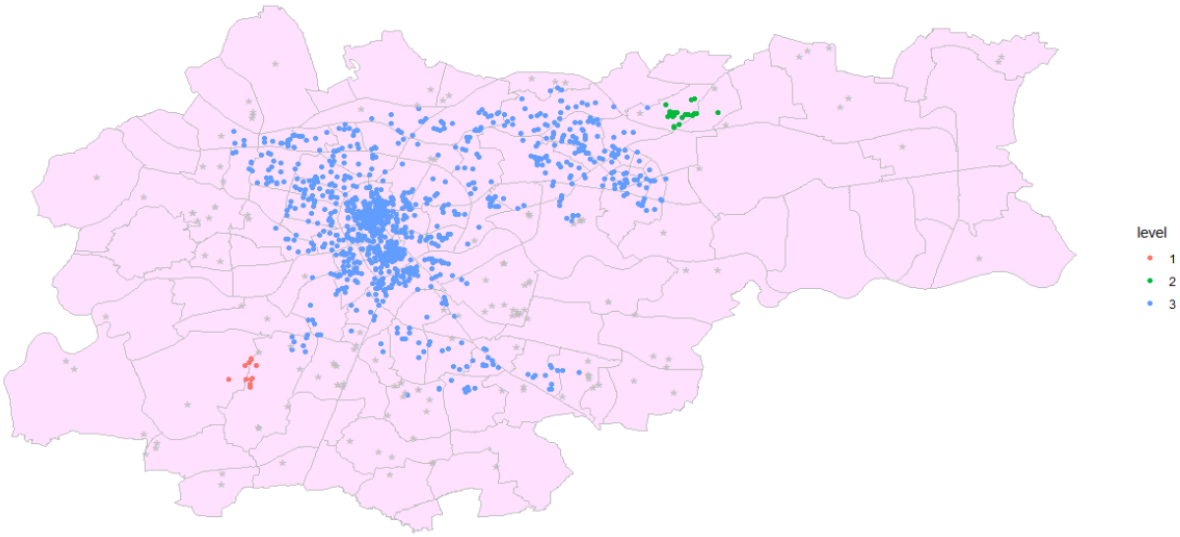


2. *minPts=4*
   Wynik bardzo słaby. Jeden klaster został utworzony przez większą część Krakowa. Mało prawdopodobne, aby wykroczenia na terenie całego Krakowa miały ze sobą tyle wspólnego.



3. *minPts=13*

   Wynik podobny do poprzedniego, daje więcej punktów szumu.



Porównanie wyników:

W tym algorytmie również wydzieliła się różna ilość klastrów dla różnego minPts. Jednakże jedyny sensowny wynik uzyskujemy przy minPts=6. Utworzyły się aż 31 klastrów, jednak biorąc pod uwagę cały obszar Krakowa i wykroczenia na tym obszarze jest to możliwe. Mało prawdopodobna jest opcja nr 2 i 3 ponieważ większość wykroczeń należy głównie do jednego klastra – to by oznaczało, że mają ze sobą dużo wspólnego, a najprawdopodobniej tak nie jest.

Dzielnice dla najlepszej opcji(1):

Większy na terenie Bieńczyc, Mistrzejowic, kawałek Nowej Huty, Czyżyn. Większy również na terenie Starego Miasta, Grzegórzki, Podgórze, Krowodrza, kawałek dzielnicy Dębniki. Mniejsze na terenie Wzgórz Krzesławickich, Bieżanów-Prokocim, Podgórze, Łagiewniki-Borek Fałęcki, Podgórze Duchackie, Dębniki, Zwierzyniec oraz Bronowice.

Wady	*Zalety*
Brak jednoznacznego kryterium pozwalającego stwierdzić, jaka jest właściwa graniczna liczebność wykrywanych klastrów	*Lepiej grupuje dane niż algorytm k-średnich*
	Daje dobre wyniki przy klastrach o dowolnym kształcie, różnych rozmiarach oraz gęstościach
	szybszy niż DBSCAN

ALGORYTM OPTICS

Działanie

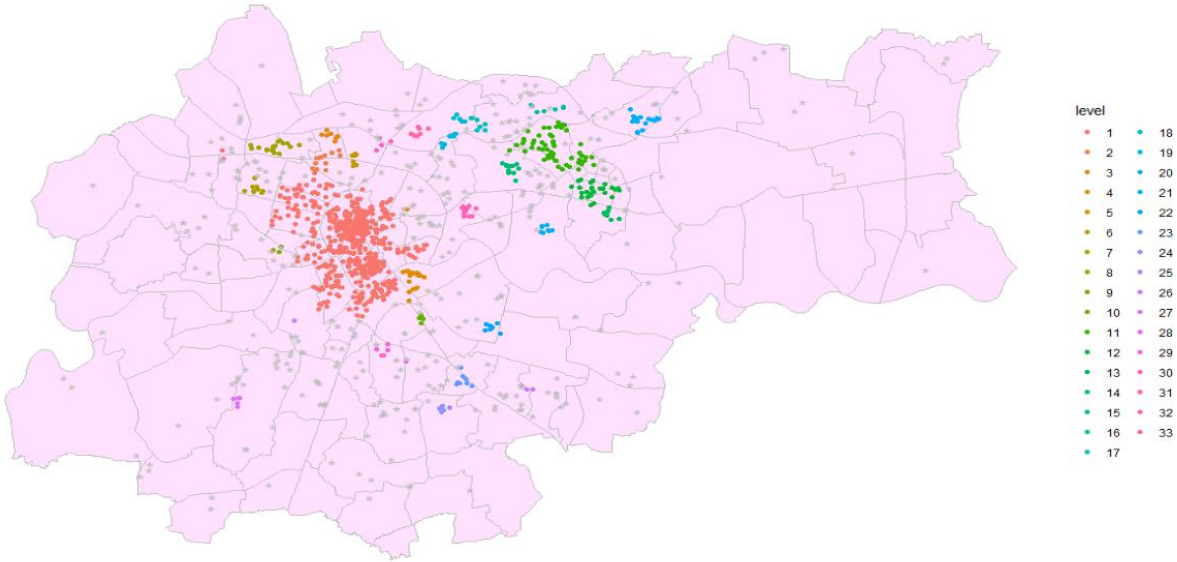
Wykorzystuje podobne koncepcje co DBSCAN. Różnicą jest to, że parametr eps to tylko górna granica promienia sąsiedztwa używana do zmniejszania złożoności obliczeniowej. Parametr minPts również działa inaczej. Używa się go do określenia gęstych sąsiedztw. Algorytm oblicza osiągalności dla każdego punktu. W porównaniu do DBSCANY ten algorytm nie jest w stanie przypisać punktów granicznych i ustawia je jako szum.

Kod i wyniki

```
library(fpc)
opt <- optics(myData, eps = 500,  minPts = 6)
opt <- extractDBSCAN(opt, eps_cl = 300)
noise<-myData[opt$cluster==0,]
data<-myData[opt$cluster!=0,]
clusters<-as.factor(opt$cluster)
level<-clusters[clusters!=0]
ggplot() +
  geom_polygon(data = spdf_fortified, aes( x = long, y = lat, group = group), fill="thistle1", color="grey") +
  theme_void() +geom_point(aes(data$Lon,data$Lat, colour = level))+geom_point(aes(noise$Lon,noise$Lat), pch="*", size=5, col="grey")
```

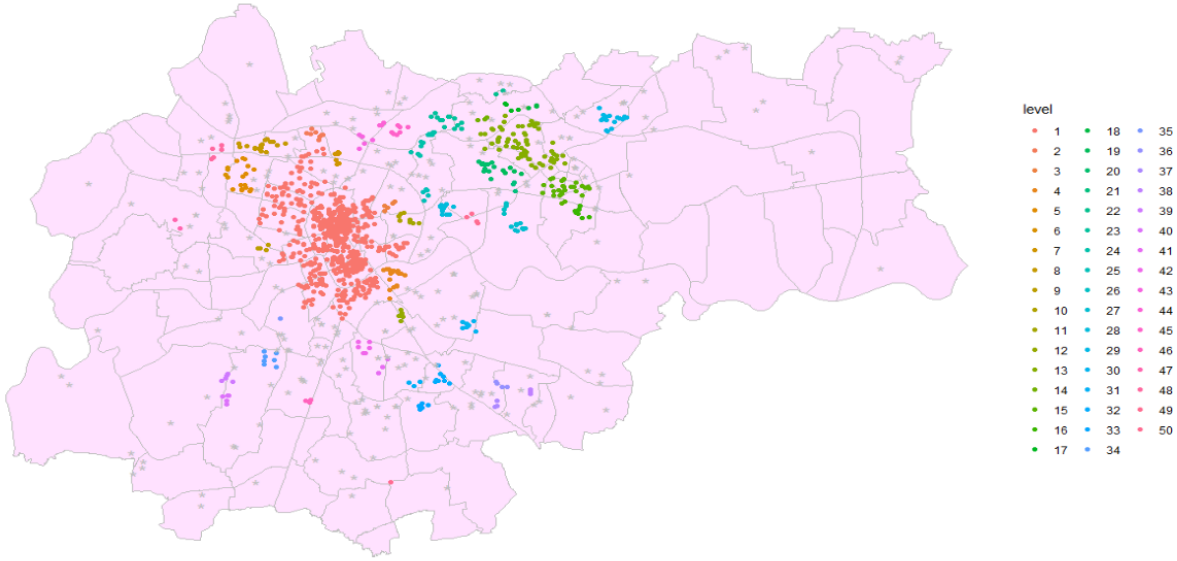
1. *eps=500, minPts=6*

   Wynik w miarę satysfakcjonujący, ale spróbuję też innych parametrów.

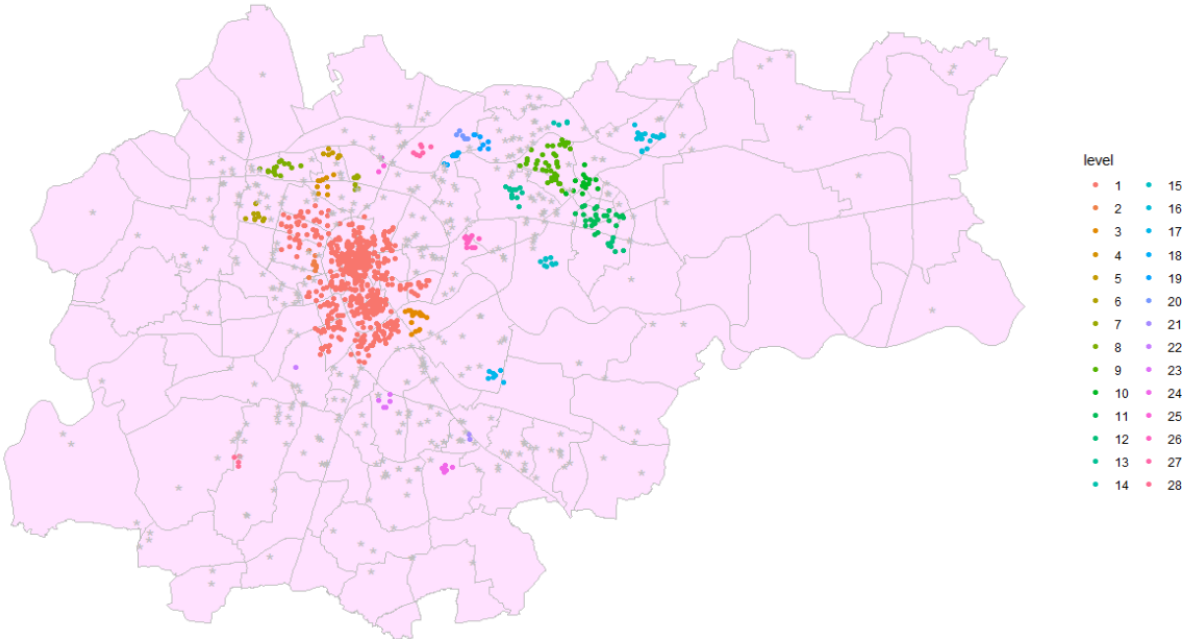


2. *eps=520, minPts=4*

   Utworzona została bardzo duża ilość klastrów.



3. *eps=510, minPts=7*
   Mniejsza ilość klastrów, wynik bardzo dobry, gęste obszary w jednym klastrze.



Porównanie wyników:

Ten algorytm moim zdaniem najlepiej podzielił wykroczenia na obszarze Krakowa na klastry. Klastry są rozdzielone stosunkowo względem danych osiedli. Zwłaszcza przy opcji nr 3 patrząc na centrum nie mamy już tak bardzo rozległego obszaru w jednym klastrze. 1 przypadek jest podobny do ostatniego, zaś w drugim mniej punktów uznano jako szum.

Dzielnice dla najlepszej opcji (3):

Większy dla Starego Miasta, Grzegórzki, Krowodrzy oraz kawałek Podgórza.

Mniejszy dla Nowej Huty i Czyżyn. Dla Dębnik, Podgórza Dychackiego oraz Bieżanów-Prokocim. Jeden na Wzgórzach Krzesławickich, Krowodrzy, Mistrzejowic oraz Prądnika Białego.

Wady	*Zalety*
Nie jest w stanie ustawić niektórych punktów granicznych i określa je jako szum	*Lepiej grupuje dane niż algorytm k-średnich*
Dobry dla różnych gęstości	*Daje dobre wyniki przy klastrach o dowolnym kształcie, różnych rozmiarach oraz gęstościach*
Trudny wybór parametrów	*szybszy niż DBSCAN*

PORÓWNANIE WYNIKÓW RÓŻNYCH ALGORYTMÓW

Porównując wszystkie algorytmy stwierdzam, że najlepszym okazał się OPTICS. Najlepiej dopasował się pod moje dane. Jest szybszy od DBSCANU i lepiej je grupuje niż HDBSCAN. W moim przypadku HDBSCAN zachowywał się nietypowo, ponieważ dla większości parametrów tworzył około 2-3 klastry co oznaczałoby, że wykroczenia na obszarze całego Krakowa mają ze sobą dużo wspólnego. Lepiej utworzyć więcej ilości klastrów, których wykroczenia będą bardziej zbliżone i podobne do siebie. Jednakże patrząc na algorytm HDBSCAN z parametrem minPts=6 wypadł lepiej niż zwykły DBSCAN – wystarczy spojrzeć na obszar centrum Krakowa. DBSCAN utworzył jeden duży klaster, gdzie w HDBSCANIE'e udało się stworzyć dodatkowe inne klastry – ta sytuacja jest bardziej prawdopodobna.